MF324

工业级- 8 位 MTP

无刷直流电机控制器

数据手册

第 **0.03** 版

2026 年 **10** 月 **21** 日

Copyright © 2026 by PADAUK Technology Co., Ltd., all rights reserved.
6F-6, No.1, Sec. 3, Gongdao 5th Rd., Hsinchu City 30069, Taiwan, R.O.C.
TEL: 886-3-572-8688 www.padauk.com.tw

重要声明

应广科技保留权利在任何时候变更或终止产品，建议客户在使用或下单前与应广科技或代理商联系以取得最新、最正确的产品信息。

应广科技不担保本产品适用于保障生命安全或紧急安全的应用，应广科技不为此类应用产品承担任何责任。关键应用产品包括，但不仅限于，可能涉及的潜在风险的死亡，人身伤害，火灾或严重财产损失。

应广科技为服务客户所提供之任何编程软件，皆为服务与参考性质，不具备任何软件漏洞责任，应广科技不承担任何责任来自于因客户的产品设计所造成的任何损失。在应广科技所保障的规格范围内，客户应设计和验证他们的产品。为了尽量减少风险，客户设计产品时，应保留适当的产品工作范围安全保障。

提供本文档的中文简体版是为了便于了解，请勿忽视文中英文的部份，因为其中提供有关产品性能以及产品使用的有用信息，应广科技暨代理商对于文中可能存在的差错不承担任何责任，建议参考本文件英文版。

目录

修订历史 **7**
警告 **7**
1. 单片机特点 **8**
1.1. 系统特性 8
1.2. 系统功能 8
1.3. 高性能 RISC CPU 架构 9
1.4. 订购/封装信息 9
2. 系统概述和方框图 **10**
3. 引脚分配及功能说明 **11**
4. 器件电气特性 **12**
4.1. 绝对最大值 12
4.2. 直流/交流特性 12
4.3. ILRC 频率与 VDD 和温度关系曲线图 14
4.4. IHRC 频率偏差与 VDD 温度关系曲线图 14
4.5. 典型 ILRC 频率与 VDD 的关系 15
4.6. 典型 IHRC 频率偏差与 VDD 的关系 15
4.7. 典型工作电流与 VDD 和 CLK=IHRC/n 的关系 16
4.8. 工作电流与 VDD、系统时钟= =ILRC/n 关系 16
4.9. 引脚上拉/下拉电阻曲线图 17
4.10. 引脚输出驱电流(I_{OH})与灌电流(I_{OL}) 曲线图 18
4.11. 引脚输入高电压与低电压（VIH/VIL）曲线图 19
5. 中央处理器(CPU) **19**
5.1. 功能描述 19
5.1.1. 处理单元 20
5.1.2. 程序计数器 22
5.1.3. 程序结构 23
5.1.4. 算术和逻辑单元 23
5.2. 存储器 24
5.2.1. 程序存储器 – ROM 24
5.2.2. 数据存储器 – SRAM 26
5.2.3. 系统寄存器 27
5.2.3.1. ACC 状态标志寄存器 (FLAG), 地址= 0x00 27
5.2.3.2. 多核使能寄存器 (MCUEN), 地址= 0x01 28
5.2.3.3. 选项寄存器 2 (OPR2)，地址 = 0x47 28
5.2.3.4. 杂项寄存器 (MISC), 地址= 0x34 28
5.3. 堆栈 29
5.3.1. 堆栈指针寄存器(*SP*), 地址= 0x02 30
5.4. 程序选项 30

6. 振荡器和系统时钟 30
6.1. 内部高频振荡器和内部低频振荡器 30
6.2. 系统时钟和 IHRC 校准 31
6.2.1. 系统时钟 31
6.2.1.1. 时钟模式寄存器(CLKMD), 地址= 0x03 31
6.2.2. 频率校准 32
6.2.2.1. 特别声明 33
6.2.3. 系统时钟切换 33
7. 复位 34
7.1. 上电复位 - POR 34
7.2. 低压复位- LVR 35
7.3. 看门狗超时溢出复位 36
7.4. 外部复位 - PRSTB 37
8. 中断 38
8.1. 中断允许寄存器 (*INTEN*), 地址= 0x04 39
8.2. 中断允许 2 寄存器 (*INTEN2*), 地址= 0x08 39
8.3. 中断请求寄存器(*INTRQ*), 地址= 0x05 40
8.4. 中断请求 2 寄存器(INTRQ2), 地址= 0x09 40
8.5. 中断缘选择寄存器 (INTEGS), 地址= 0x0C 41
8.6. 中断工作流程 41
8.7. 中断的一般步骤 42
8.8. 使用中断举例 43
9. I/O 端口 44
9.1. IO 相关寄存器 44
9.1.1. 端口 A 数字输入启用寄存器(PADIER)，地址= 0x0D 44
9.1.2. 端口 A 数据寄存器(PA)，地址 = 0x0F 44
9.1.3. 端口 A 控制寄存器(PAC), 地址 = 0x10 44
9.1.4. 端口 A 上拉控制寄存器(PAPH), 地址= 0x11 44
9.1.5. 端口 A 下拉控制寄存器 (PAPL), 地址= 0x12 44
9.1.6. 端口 B 数据寄存器 (PB), 地址= 0x13 44
9.1.7. 端口 B 控制寄存器(PBC), 地址= 0x14 44
9.2. IO 结构及功能 45
9.2.1. IO 引脚的结构 45
9.2.2. IO 引脚的一般功能 46
9.2.3. IO 的使用与设定 46
10. Timer / PWM 计数器 47
10.1. 16 位计数器 (Timer16) 47
10.1.1. Timer16 介绍 47
10.1.2. Timer16 溢出时间 48
10.1.3. Timer16 控制寄存器(*T16M*)，地址 = 0x06 49
10.2. 16-bit Timer (Timer4) 50

10.2.1. Timer4 介绍 50
10.2.2. Timer4 模式寄存器（*TM4C*），地址=0x22 51
10.2.3. Timer4 上限高字节寄存器(TM4CTH)，地址=0x23 51
10.2.4. Timer4 上限低字节寄存器（TM4CTL），地址 = 0x24 51
10.2.5. Timer4 高字节寄存器（TM4BH），地址= 0x25 51
10.2.6. Timer4 低字节寄存器（TM4BL），地址= 0x26 51
10.3. 8 位计数器（Timer2, Timer3) 52
10.3.1. Timer2 模式寄存器（*TM2C*），地址= 0x27 53
10.3.2. Timer2 高字节寄存器（*TM2CT*），地址= 0x28 53
10.3.3. Timer2 分频寄存器（*TM2S*），地址= 0x29 53
10.3.4. Timer2 上限寄存器 （*TM2B*），地址 = 0x2A 53
10.3.5. Timer3 控制寄存器 （*TM3C*），地址 = 0x2B 53
10.3.6. Timer3 上限寄存器（*TM3CT*），地址= 0x2C 54
10.3.7. Timer3 分频寄存器（*TM3S*），地址= 0x2D 54
10.3.8. Timer3 上限寄存器(*TM3B*)，地址= 0x2E 54
10.4. 12 位 PWM 产生器 54
10.4.1. 带死区 PWM 硬件和时序框图 54
10.4.2. 时序图 55
10.4.3. 12 位 PWM 计算公式 55
10.4.4. 12 位 PWM 相关寄存器 56
10.4.4.1. PWM PB 控制寄存器 1(PWMPBC1)，地址= 0x18 56
10.4.4.2. PWM PB 控制寄存器 2(PWMPBC2)，地址= 0x19 56
10.4.4.3. PWM 发生器控制寄存器 1（PWMGC1）地址 = 0x1A 57
10.4.4.4. PWM 发生器控制寄存器 2 (PWMGC2)，地址= 0x1B 57
10.4.4.5. PWM 发生器控制寄存器（PWMGC)，地址 = 0x1C 58
10.4.4.6. 杂项设置寄存器（MISC)，地址= 0x34 58
10.4.4.7. PWM 计数器上限高位寄存器（PWMUPBH)，地址= 0x3D 58
10.4.4.8. PWM 计数器上限低位寄存器(PWMUPBL)，地址 = 0x3E 58
10.4.4.9. PWMG0 占空比高位寄存器（PWM0DTH)，地址 = 0x3F 59
10.4.4.10. PWMG0 占空比低位寄存器(PWM0DTL)，地址 = 0x40 59
10.4.4.11. PWMG ADC 触发高位寄存器（PWMADCH)，地址 = 0x41 59
10.4.4.12. PWMG ADC 触发低位寄存器（PWMADCL)，地址 = 0x42 59
10.4.4.13. PWM 前置死区寄存器(PWMDDZVF)，地址 = 0x43 59
10.4.4.14. PWM 后置死区寄存器（PWMDDZVR)，地址= 0x44 59
11. 省电与掉电 59
11.1. 省电模式（“stopexe”) 60
11.2. 掉电模式（“stopsys”） 61
11.3 唤醒 61
12. 特殊功能 62
12.1. 通用比较器 62
12.1.1. 通用比较器硬件框图 62
12.1.2. 通用比较器控制寄存器(GPCC)，地址= 0x15 63
12.1.3. 通用比较器结果寄存器(GPCR)，地址= 0x16 63

12.1.4. 模拟讯号输入 64
12.1.5. 内部参考电压 ($V_{internal\ R}$) 65
12.1.6. 使用比较器 67
12.2. 模拟-数字转换器 (ADC) 模块 68
12.2.1. AD 转换的输入要求 69
12.2.2. ADC 时钟选择 69
12.2.3. AD 转换 70
12.2.4. 配置模拟引脚 70
12.2.5. 使用 ADC 71
12.2.6. ADC 相关寄存器 72
12.2.6.1. ADC 数据高位寄存器(ADCRH), 地址 = 0x1E 72
12.2.6.2. ADC 数据低位寄存器(ADCRL), 地址 = 0x1F 72
12.2.6.3. ADC 控制寄存器 (ADCC)，地址 = 0x20 72
12.2.6.4. ADC 模式寄存器 (ADCM)，地址 = 0x21 73
12.3. PWM 生成器触发 AD 转换 73
12.3.1. PWM 触发 ADC - PWM 计数器高电平寄存器 (*PWMADCH*), 地址= 0x41 74
12.3.2. PWM 触发 ADC - PWM 计数器低电平寄存器 (PWMADCL), 地址= 0x42 74
12.4. 输入脉冲捕获 74
12.4.1. 脉冲捕获控制寄存器 (*PLSCC*)，地址 = 0x32 75
12.4.2. 脉冲捕获分频寄存器(*PLSCS*)，地址 = 0x33 75
12.4.3. 脉冲捕获脉宽高位寄存器 (*PLSPWH*)，地址 = 0x39 75
12.4.4. 脉冲捕获脉宽低位寄存器 (PLSPWL)，地址 = 0x3A 76
12.4.5. 脉冲捕获高准位脉宽高位寄存器 (*PLSPHH*)，地址 = 0x3B 76
12.4.6. 脉冲捕获高准位脉宽低位寄存器 (*PLSPHL*)，地址 = 0x3C 76
12.5. 特殊比较器 76
12.5.1. 限流比较器 76
12.5.2. 限流设置寄存器 (LCS), 地址= 0x2F 76
12.5.3. 过零点比较器 77
12.5.4. 过零点控制寄存器 (ZCPC), 地址= 0x30 77
12.5.5. 过零点设置寄存器 (ZCPS), 地址= 0x31 78
12.6. 乘法器 78
12.6.1. 8×8 乘法器 78
12.6.2. 算术运算寄存器 (*EARITH*)，地址 = 0x1D 79
12.6.3. 8X8 乘法器运算对象 1 寄存器(*M8OP1*)，地址 = 0x35 79
12.6.4. 8X8 乘法器运算对象 2 寄存器(*M8OP2*)，地址 = 0x36 79
12.6.5. 8X8 乘法器结果 1 寄存器(*M8RS1*)，地址 = 0x37 79
12.6.6. 8X8 乘法器结果 0 寄存器(*M8RS0*)，地址 = 0x38 79
13. 烧录方法 79
14. 指令 83
13.1. 指令表 84

修订历史

修 订	日期	描 述
0.00	2026/01/16	初版
0.01	2026/06/02	1. 第 11.2.2.节:ADC 时钟选择：将 ADC 时钟周期从 0.5us 修改为 0.2us 2. 第 11.2.2.节:更新图 27：ADC 框图 3. 第 11.5.1.节:限流比较器：将 PA4 / PADIER.4 修改为 PA7/ PADIER.7
0.02	2026/09/09	1. 新增第 11 节：省电与掉电 2. 新增第 11.1 节：省电模式 3. 新增第 11.2 节：掉电模式 4. 新增第 11.3 节：唤醒
0.03	2026/10/21	1. 第 6.1 节：于 IHRC 章节处，新增 PWM 关闭顺序及注意事项。 2.第 10.4 节：于 PWM 章节处，新增关闭 IHRC 或进入 STOPSYS 的 PWM 使用注意事项。

警告

在使用 IC 前，请务必认真阅读 MF324 相关的 APN（应用注意事项）。

APN 下载地址为：

https://www.padauk.com.tw/cn/product/search_list.aspx?kw=MF

1. 单片机特点

1.1. 系统特性

- 高抗干扰系列:
  特别适用于 AC 电源供电的、阻容降压电路的、需要较强抗干扰能力的，或有高 EFT 安规测试要求(±4KV)的产品。
- 工作温度范围: -40°C ~ 105°C

1.2. 系统功能

- 3KW MTP 程序空间供 8 个 FPPA 单元使用（可编程 1000 次）
- 192 字节数据空间用于所有 FPPA 单元
- 两个硬件 16 位定时器
- 两个硬件 8 位定时器
- 一个 12 位硬件 PWM 生成器，可编程设定周期和占空比
- 支持 PWM 发生器触发 ADC 转换
- 高达 11 通道 12 位 ADC:
  - 1 通道用于内部带隙基准电压
  - 1 通道用于内部 VDD/4 电压
  - 1 通道用于内部 BEMF
- 一个 BEMF 探测器，带去间隙功能
- 内置 BEMF 比较器对应引脚 PA0, PA1, PA2
- 一个通用比较器
- 一个硬件脉冲捕捉器
- 一个 8X8 硬件乘法器
- 支持三相无刷直流电机
- 7 个 IO 引脚具有 I_{OH} -和 I_{OL} 10mA 功能，可选上拉/下拉电阻器
- 1 个 IO 引脚，支持 I_{OL} 10mA，且可选配上拉电阻
- 6 个专用 PWM 引脚，具有 I_{OH} -10mA 和 I_{OL} 20 mA 功能，可选择下拉电阻
- 16 级 VDD 电压复位
- 可选的三个外部中断引脚：PA3, PA4, PA6
- 工作电压范围：2.2V ~ 5V

1.3. 高性能 RISC CPU 架构

- 现场可编程处理器陈列(FPPA™)技术专利
- 工作模式：八个 FPPA™处理单元运作模式
- 87 条高效的指令
- 绝大部分指令都是单周期(1T)指令
- 可程序设定的堆栈指针和堆栈深度
- 数据存取支持直接和间接寻址模式，用数据存储器即可当作间接寻址模式的数据指针(index pointer)
- 提供可保护 MTP 资料的加密功能
- 寄存器空间、数据存储空间、MTP 程序空间三者互相独立

1.4. 订购/封装信息

- MF324-1J16A：QFN(3*3*0.75mm)
- MF324-S16A：SOP16(150mil)

- 有关封装尺寸的信息，请参阅官方网站文件："封装信息"

2. 系统概述和方框图

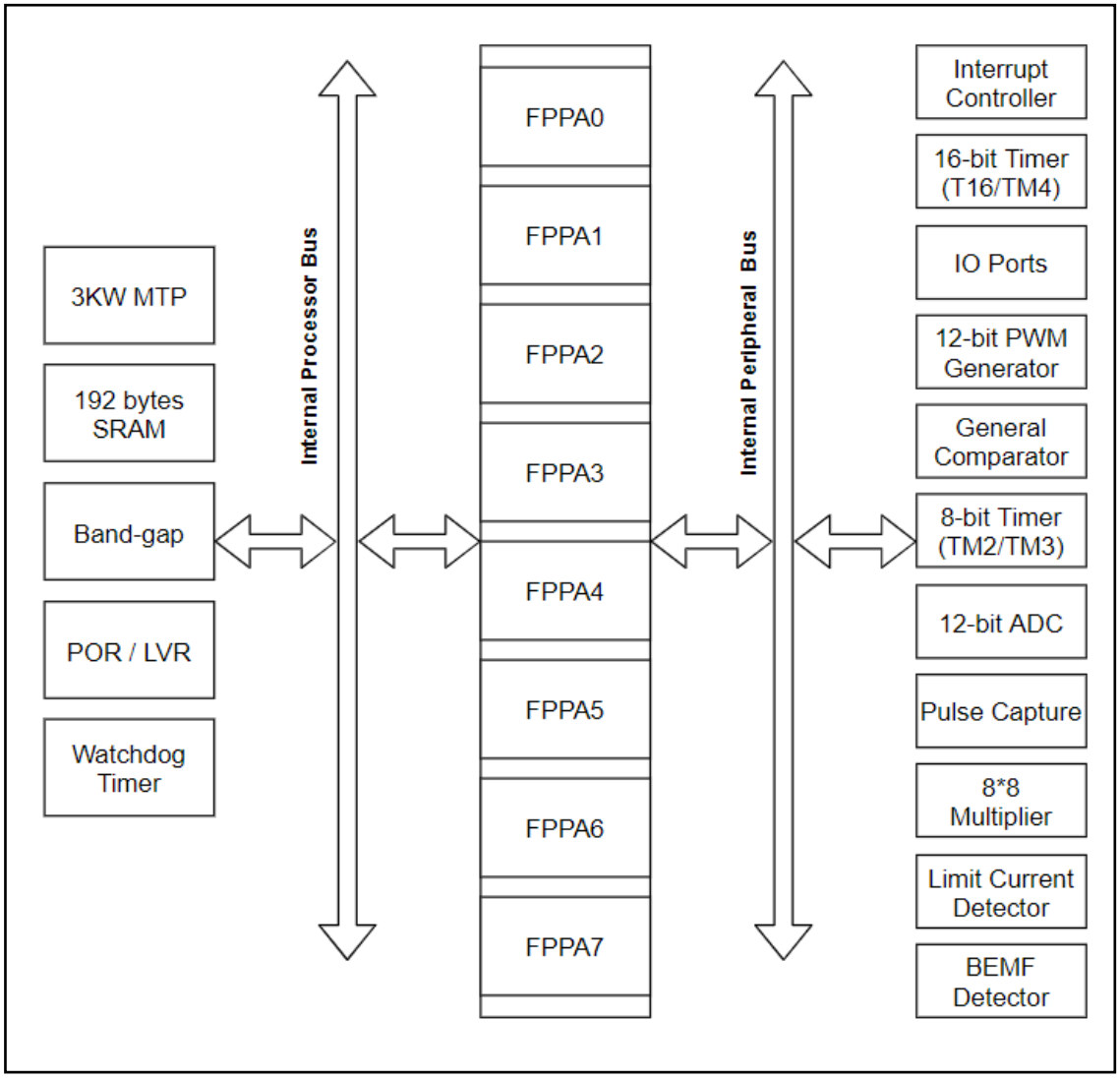
MF324 是应广科技（PADAUK）推出的一款基于 MTP 技术的三相无刷直流电机（3-Phase BLDC）专用处理器，采用并行处理架构与全静态 CMOS 8×8 位处理器阵列设计，可并行执行八项外设功能。该芯片基于专利申请中的 FPPA™（现场可编程处理器阵列）技术，采用精简指令集计算机（RISC）架构，除处理间接存储器访问的部分指令需两个周期外，其余所有指令均为单周期执行。

芯片内置 3KB MTP 程序存储器与 192 字节数据静态随机存取存储器（SRAM），供 8 个 FPPA 单元共享使用。

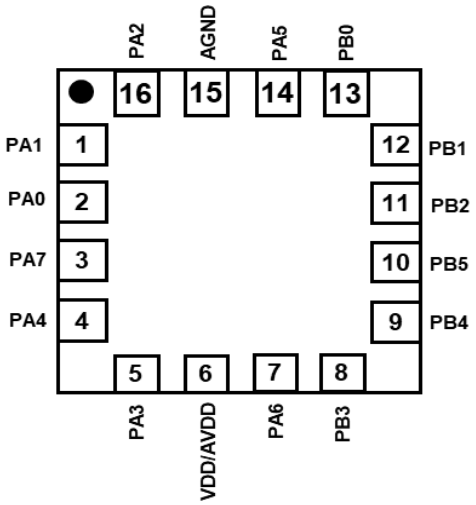
MF324 内置 11 通道 12 位分辨率 A/D 转换器。

MF324 提供一个通用比较器和四个硬件定时器： 其中两个是 16 位定时器，两个是 8 位定时器。

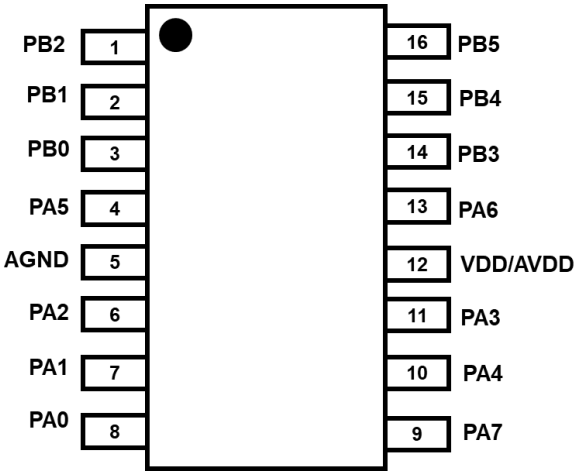
此外，MF324 还内置了一个硬件脉冲捕获器、一个 BEMF 检测器、一个 12 位硬件 PWM 发生器和 PWM 保护模块，为无刷直流控制器提供最佳解决方案。



3. 引脚分配及功能说明



MF324-1J16A: QFN(3*3*0.75mm)



MF324-S16A: SOP16(150mil)

引脚说明:

引脚名称	输入/ 输出			特殊功能						
	I / O	上拉/下拉	唤醒	GPC BEMF/LC	GPC	PWM	脉冲捕获	ADC	外部中断/复位	烧录
PA0	I / O	H / L		BEMF-	CMP-			AD0		
PA1	I / O	H / L		BEMF-	CMP-			AD1		
PA2	I / O	H / L		BEMF-	CMP-			AD2		
PA3	I / O	H / L	V		CMP-		PCIN	AD3	INT0	Program
PA4	I / O	H / L	V	LC Flag	CMP-		PCIN	AD4	INT1	
PA5	I	H	V	BEMF-	CMP+			AD5	RST	Program
PA6	I / O	H / L	V	BEMF Flag	CMP Flag			AD6	INT2	Program
PA7	I / O	H / L	V		CMP-		PCIN	AD7		
PB0	O					V				
PB1	O					V				
PB2	O					V				
PB3	O					V				
PB4	O					V				
PB5	O					V				
VDD / AVDD										Program
GND / AGND										Program
注意	1. 所有 I/O 引脚都具有：施密特触发器输入；CMOS 电压基准位。 2. 当某引脚作为 PWM 输出端口时，其 IO 功能自动停用。 3. 当 PA5 引脚设定成输入时，对于需要高抗干扰能力的系统，请串接 33Ω 电阻。									

4. 器件电气特性

4.1. 绝对最大值

名称	最小值	典型值	最大值	单位	备 注
电源电压 (VDD)	2.2	5.0	5.5	V	电源电压不能超过最大值，否则可能损坏 IC
工作温度	-40		105	°C	
储藏温度	-50		125	°C	
节点温度		150		°C	

4.2. 直流/交流特性

符号	特性	最小值	典型值	最大值	单位	条件(Ta=25°C)
LVR%	低电压复位公差	-10		10	%	
f_{SYS}	系统时钟 (CLK)* = IHRC/2		8M		Hz	V_{DD} ≧ 2.75V@105 °C V_{DD} ≧ 2.5V@35 °C
	IHRC/4 IHRC/8 内部低频 RC 振荡器		4M 2M 45K			V_{DD} ≧ 2.0V V_{DD} ≧ 1.8V V_{DD} = 5.0V
P_{cycle}	烧录次数	1000			cycles	
V_{POR}	上电复位电压		2.1*		V	* 受限于 LVR 公差
I_{OP}	工作电流		2 1.5 0.4		mA	f_{SYS}=8MIPS@5.0V f_{SYS}=4MIPS@5.0V f_{SYS}=ILRC= 48KHz@5.0V
I_{PD}	掉电模式下电流 （使用 ***stopsys*** 命令）		0.6		uA	V_{DD}=5.0V; 仅 ILRC 模块处于开启状态
I_{PS}	省电模式下电流 （使用 ***stopexe*** 命令）		8		uA	V_{DD}=5.0V; Bandgap, LVR, IHRC, ILRC, Timer16 模块处于开启状态
V_{IL}	输入低电压	0		0.2VDD	V	
V_{IH}	输入高电压	0.8VDD 0.7VDD		VDD	V	PA5 Other IO
I_{OL}	**IO 输出灌电流**					
	PB0 – PB5 PA5 其他引脚		47 X 16		mA	V_{DD}=5.0V, V_{OL}=0.5V
I_{OH}	**IO 输出驱动电流**					
	PB0 – PB5 PA5 其他引脚		18 X 12		mA	V_{DD}=5.0V, V_{OH}=4.5V
R_{PH}	上拉电阻		120 71		KΩ	PA5 V_{DD}=5.0V, Other IO
R_{PL}	下拉电阻		X 71		KΩ	PA5 V_{DD}=5.0V, Other IO
V_{BG}	Bandgap 参考电压	1.12	1.20	1.28	V	V_{DD}=5V -40°C <Ta<105°C*

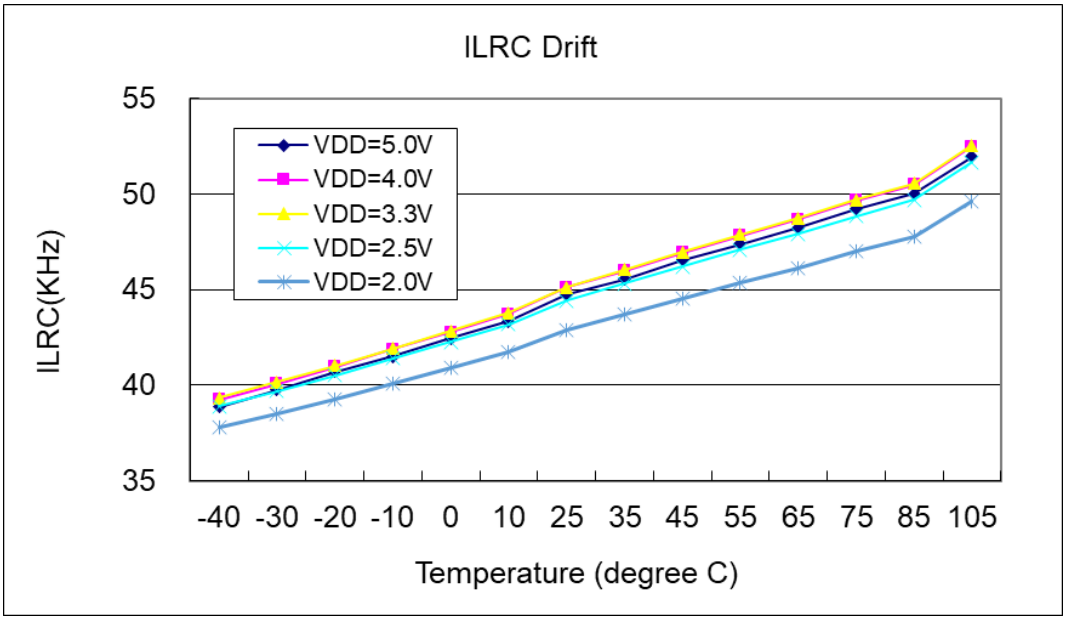
©Copyright 2026, PADAUK Technology Co. Ltd

符号	特性	最小值	典型值	最大值	单位	条件(Ta=25°C)
f_{IHRC}	校准后 IHRC 频率 *	15.52*	16*	16.48*	MHz	25°C, V_{DD}=3.15V~5.5V
		14*	16*	17.28*		V_{DD}=3.15V~5.5V, -40°C <Ta<105°C*
V_{ADC}	ADC 可工作电压	2.4		VDD	V	V_{DD}=5V
V_{AD}	AD 输入电压	0		$ADCV_{REF}$	V	
ADrs	ADC 分辨率			12	bit	
ADclk	ADC 时钟周期		2		us	V_{ADC}=2.4V ~ VDD
t_{ADCONV}	ADC 转换时间（T_{ADCLK} 是选定 AD 转换时钟周期）		16		T_{ADCLK}	
AD DNL	ADC 微分非线性		±3*		LSB	
AD INL	ADC 积分非线性		±3*		LSB	
ADos	ADC 失调电压*		3		LSB	-40°C <Ta<105°C*
t_{INT}	中断脉冲宽度	30			ns	V_{DD} = 5.0V
V_{DR}	数据存储器数据保存电压*	1.5			V	待机模式下
t_{WDT}	看门狗超时溢出时间 (T_{ILRC} 是 ILRC 的时钟周期)		4096		T_{ILRC}	misc[1:0]=01
			16384			misc[1:0]=10
t_{SBP}	系统开机时间		2600		T_{ILRC}	T_{ILRC} 是 ILRC 时钟周期
系统唤醒时间						
CPos	比较器偏置电压*	-	±10	±20	mV	
CPcm	比较器共模输入*	0		V_{DD}-1.5	V	
CPspt	比较器响应时间**		100	500	ns	上升沿和下降沿相同
CPmc	比较器模式改变所需的稳定时间		2.5	7.5	us	
LCP_{thr}	OCP 阈值	50	100	210	mV	限流比较器

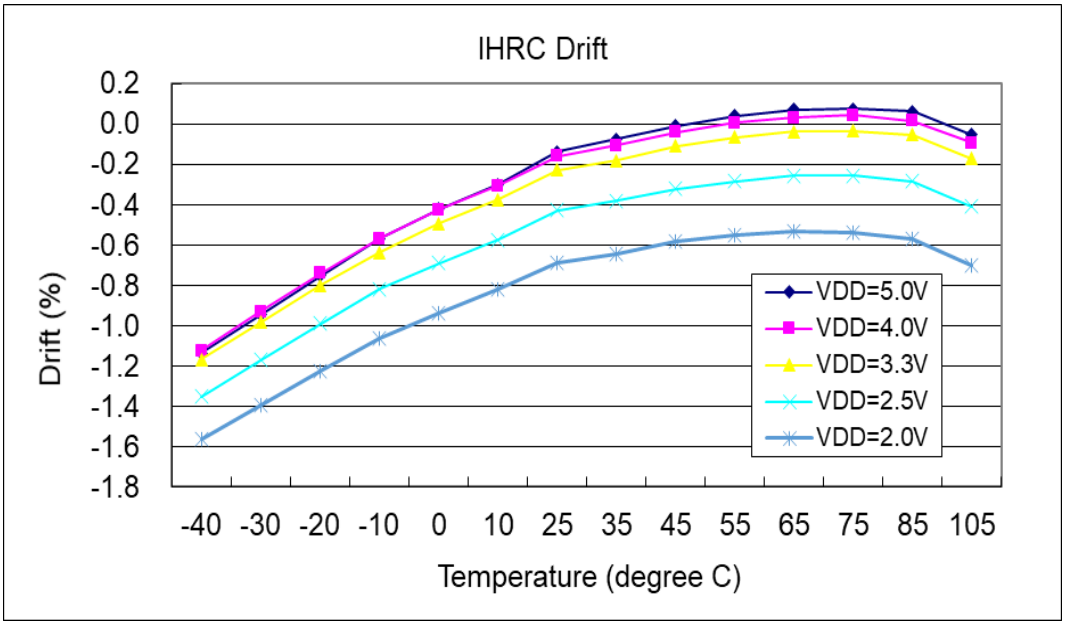
* 这些参数是设计参考值，并不是每个芯片测试。

** 特性图是实际测量值。考虑到生产飘移等因素的影响，表格中的数据是在实际测量值的安全范围内。

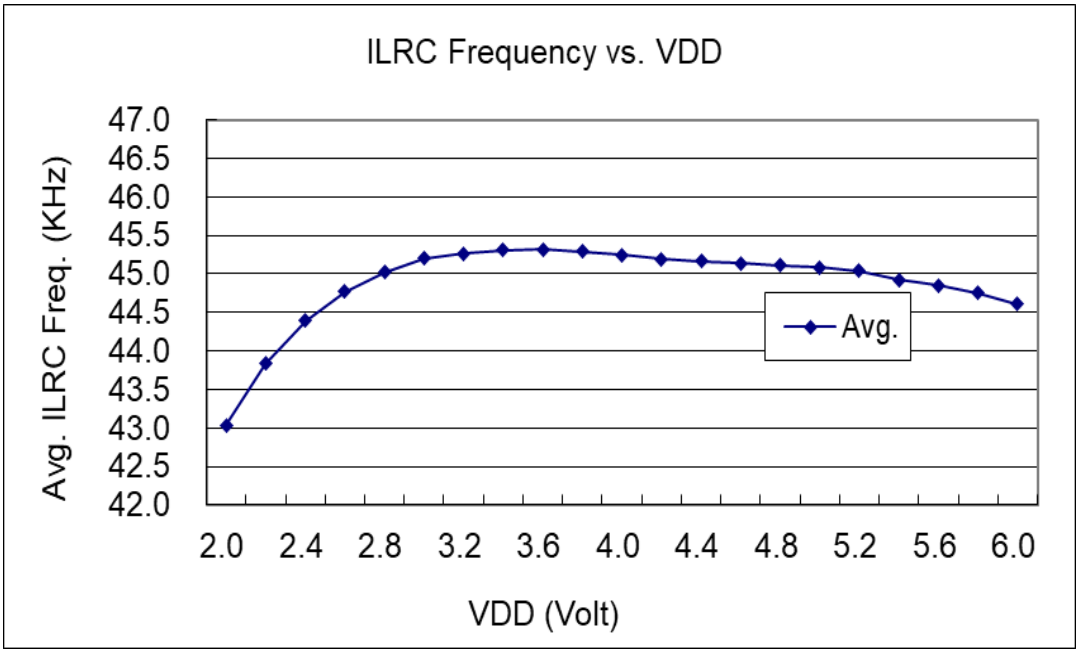
4.3. ILRC 频率与 VDD 和温度关系曲线图



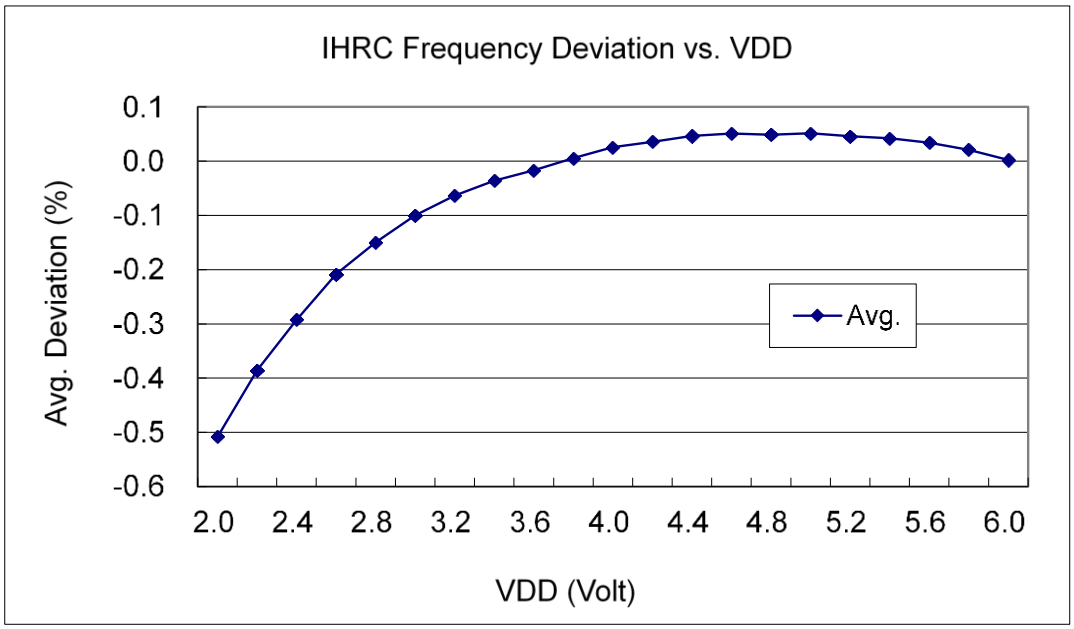
4.4. IHRC 频率偏差与 VDD 温度关系曲线图



4.5. 典型 ILRC 频率与 VDD 的关系



4.6. 典型 IHRC 频率偏差与 VDD 的关系

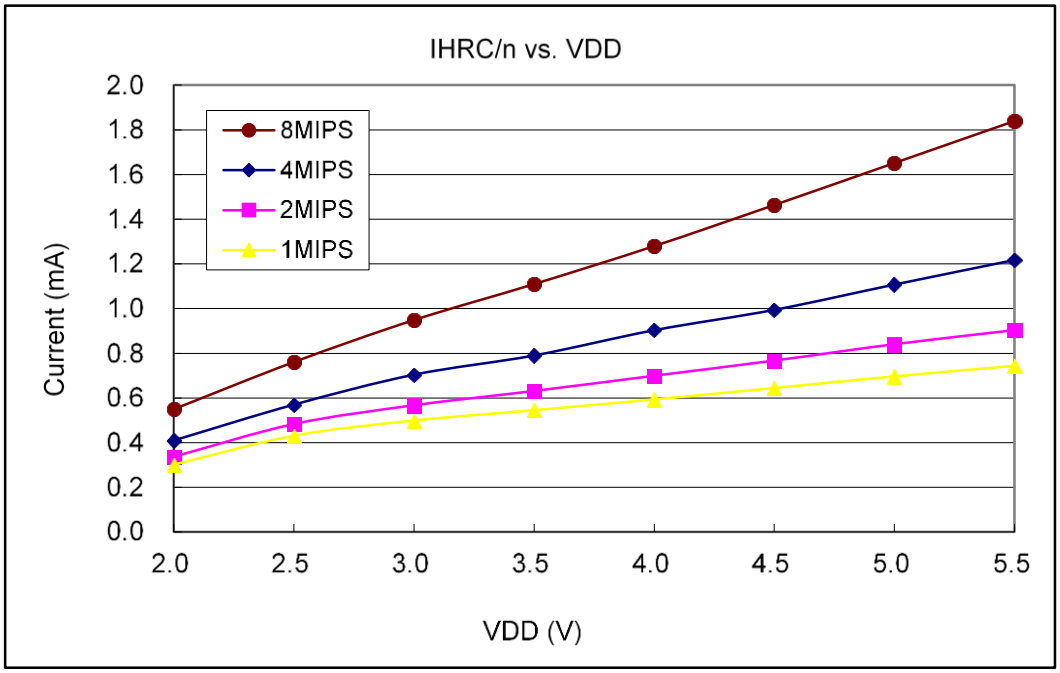


4.7. 典型工作电流与 VDD 和 CLK=IHRC/n 的关系

条件 1-FPPA （代码选项）

ON：Bandgap、LVR、IHRC；**OFF**：ILRC、T16、TM2、ADC、PWM；

IO：PA0:0.5Hz 输出切换且无负载，**其他引脚**：设为输入且不浮空

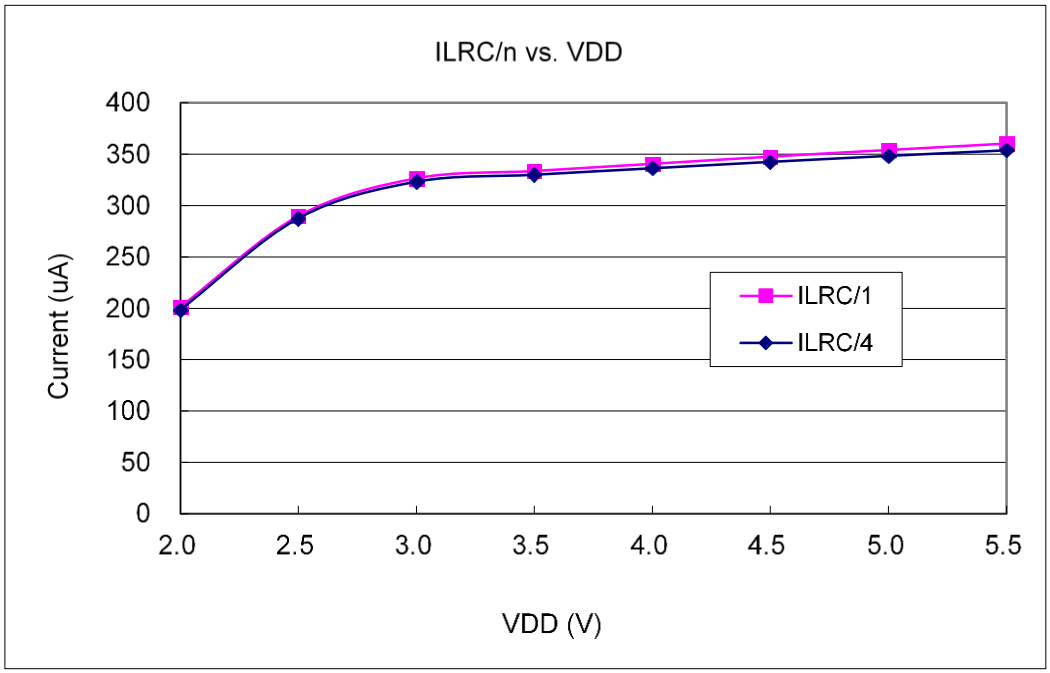


4.8. 工作电流与 VDD、系统时钟= =ILRC/n 关系

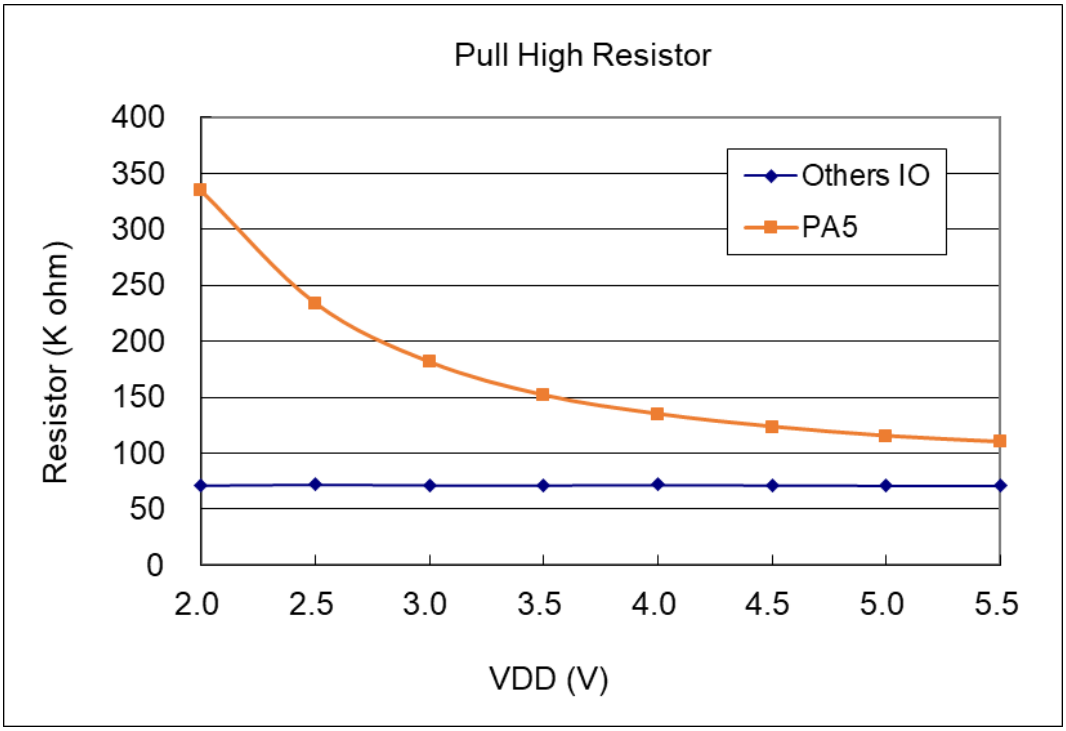
条件 1-FPPA （代码选项）

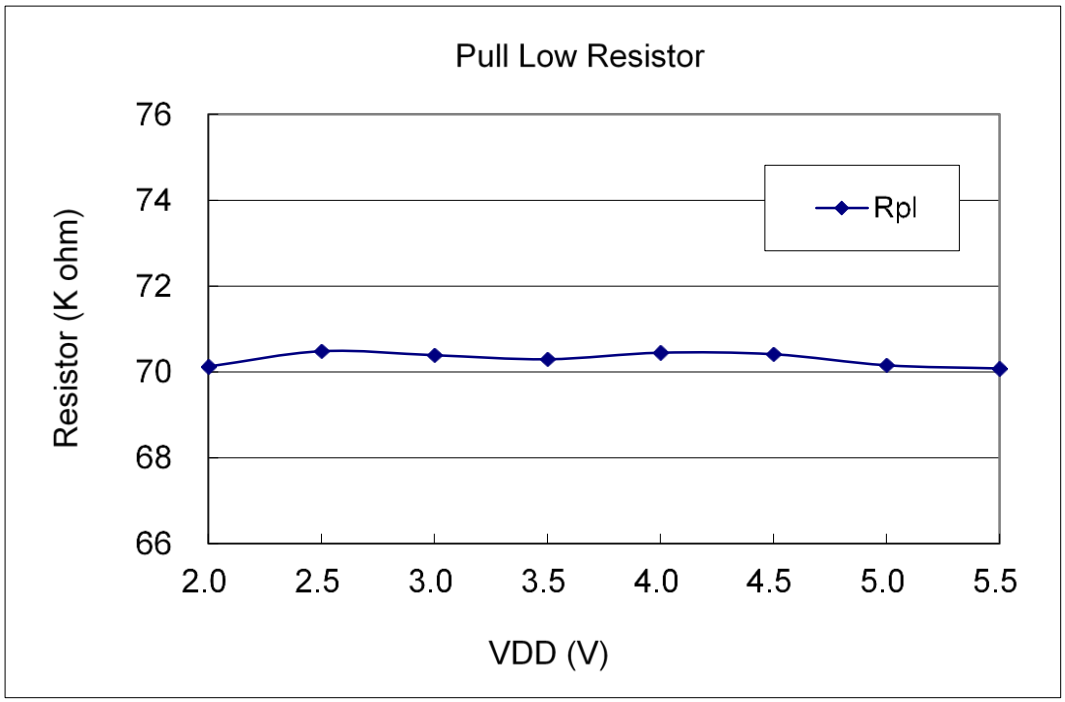
ON：Bandgap、LVR、IHRC；**OFF**：ILRC、T16、TM2、ADC、PWM；

IO：PA0:0.5Hz 输出切换且无负载，**其他引脚**：设为输入且不浮空



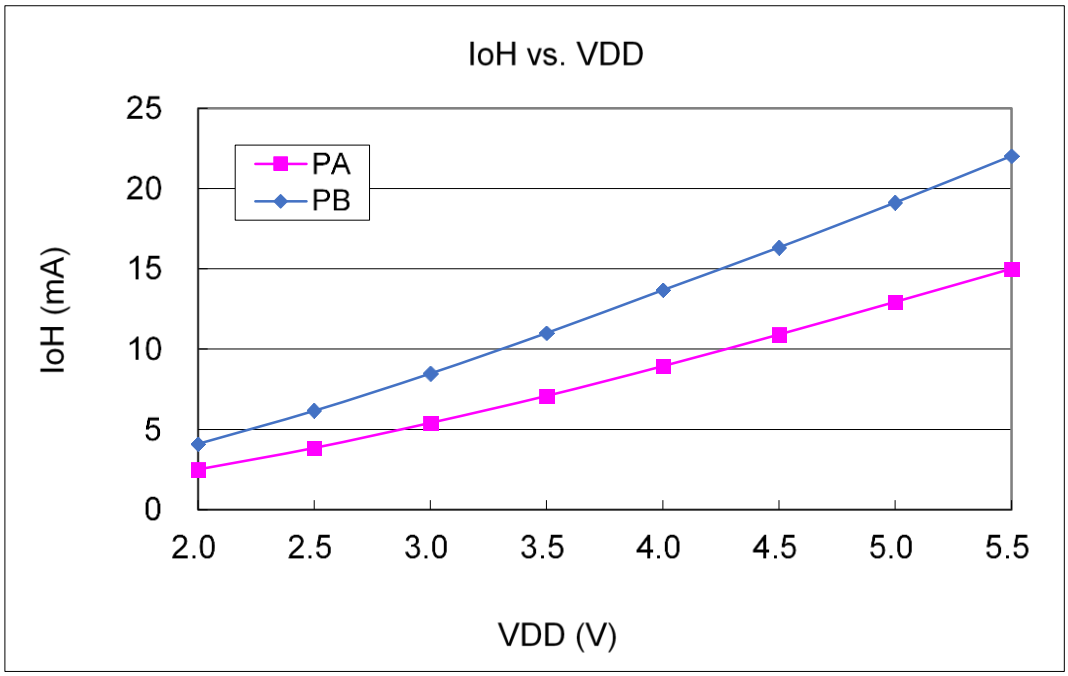
4.9. 引脚上拉/下拉电阻曲线图

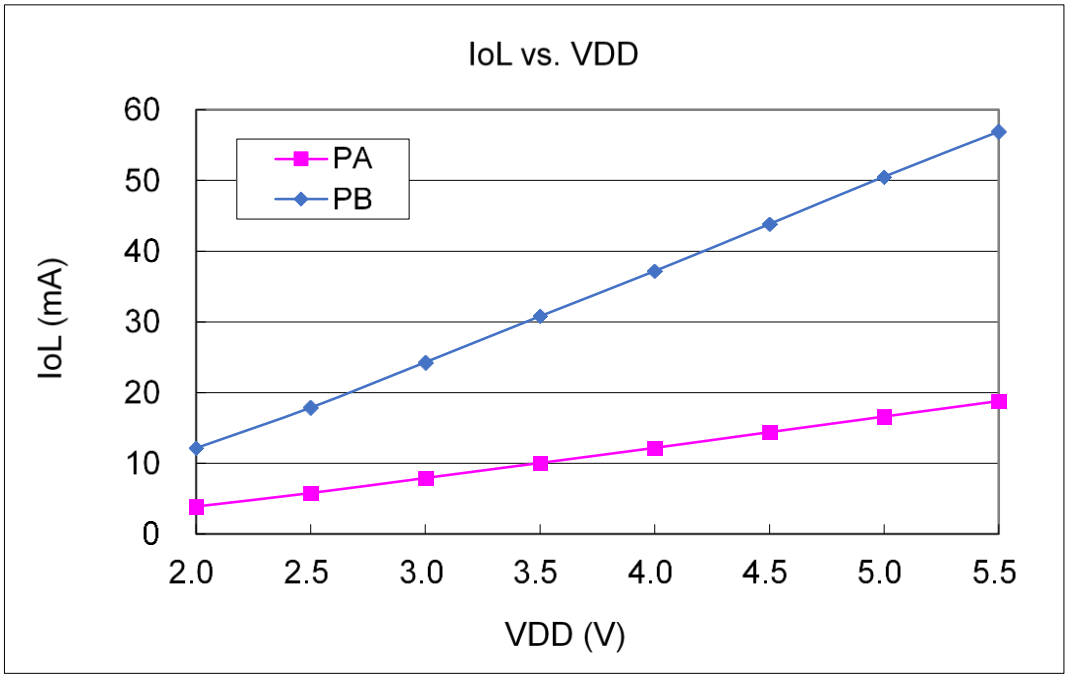




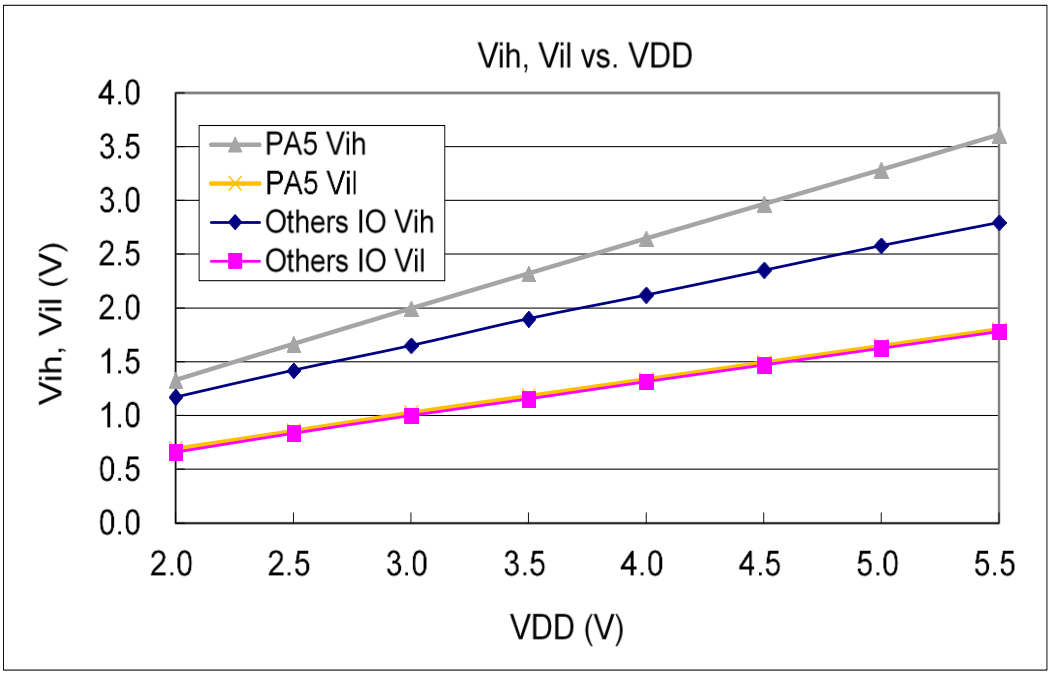
4.10. 引脚输出驱电流(I_{OH})与灌电流(I_{OL}) 曲线图

(VOH=0.9*VDD, VOL=0.1*VDD)





4.11. 引脚输入高电压与低电压（VIH/VIL）曲线图



5. 中央处理器(CPU)

5.1. 功能描述

MF324 内有八个处理单元，在每一个处理单元中包括：

a. 其本身的程序计数器来控制程序执行的顺序
b. 自身的堆栈指针用来存储或恢复程序计数器的程序执行
c. 自身的累加器
d. 状态标志以记录程序执行的状态。

每一个 FPPA 都有自己的程序计数器和累加器用以执行程序，标志寄存器以记录程序状态，堆栈指针做为跳跃操作。基于这样的架构，每个 FPPA 可以独立执行自己程序，达到并行处理效能。

5.1.1. 处理单元

八个 FPPA 单元共享相同的 3Kx15 位用户程序存储器、192 字节数据 SRAM 和所有 IO 端口，这八个 FPPA 单元在互斥的时钟周期内运行，以避免干扰。芯片内部有一个工作切换硬件模块以决定不同 FPPA 相对应的周期。图 1 所示为 FPPA 单元硬件框图。

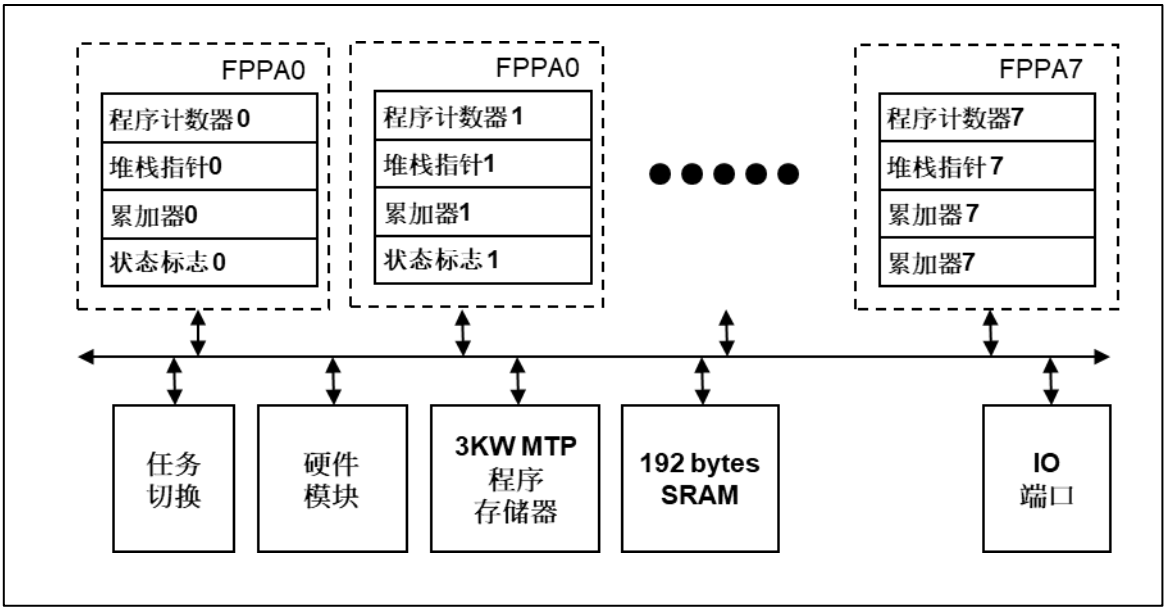


图 1：FPPA 单元结构

这八个 FPPA 单元各自独立运作在相斥的时钟周期，可以独立工作。

通过 *pmode* 命令可将系统性能共享给指定的 FPPA 单元；请参阅 *pmode* 说明。带宽分配与 FPPA 启用无关，即使 FPPA 被禁用，带宽也会被分配给指定的 FPPA 单元。

两个处理单元工作模式

pmode=0 时，会将带宽仅分配给两个 FPPA 单元，时序图如图 2 所示。每个 FPPA 单元具有整个系统一半的计算能力，例如，如果系统时钟为 8MHz，FPPA0 和 FPPA1 将分别在 4MHz 时钟下工作。

对于 FPPA0 而言，其程序将按顺序每两个系统时钟执行一次，如图：FPPA0 在第（M-1），第 M，……第（M+4）时钟周期执行程序。对于 FPPA1 而言，其程序将按顺序每两个系统时钟执行一次，如图：FPPA1 在第（N-1），第 N，……第（N+3）时钟周期执行程序。

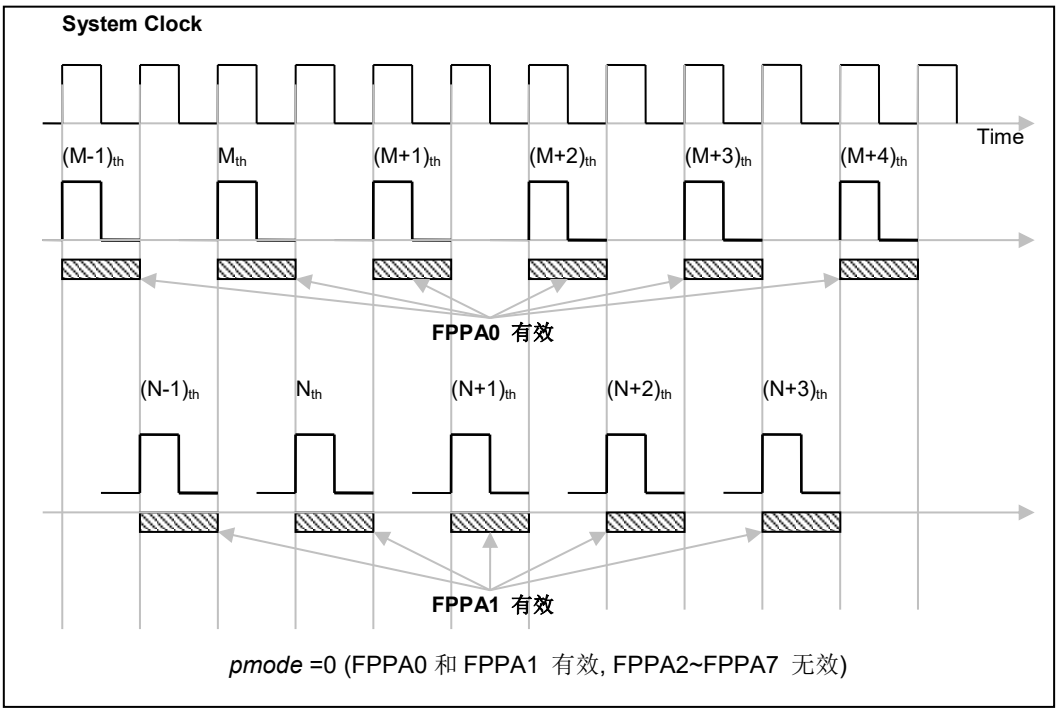


图 2：*pmode*=0 时处理单元时序图

四个处理单元工作模式

pmode=6 时，会将带宽分配给四个 FPPA 单元 (FPPA0, FPPA1, FPPA2, FPPA3)，时序图如图 3 所示。每个 FPPA 单元具有整个系统四分之一的计算能力，例如，如果系统时钟为 8MHz，FPPA0、FPPA1、FPPA2 和 FPPA3 将分别在 2MHz 时钟下工作。此时，FPPA4、FPPA5、FPPA6 和 FPPA7 不工作。

对于 FPPA0、FPPA1、FPPA2 和 FPPA3 而言，其程序将按顺序每四个系统时钟执行一次，如图：FPPA0 在第（M-1），第 M，……第（M+4）时钟周期执行程序；FPPA1 在第（N-1），第 N，……第（N+3）时钟周期执行程序；FPPA2 在第(O-1)，第 O，……第(O+3)时钟周期执行程序；FPPA3 在第(P-1)，第(P)，……第(P+3)时钟周期执行程序。

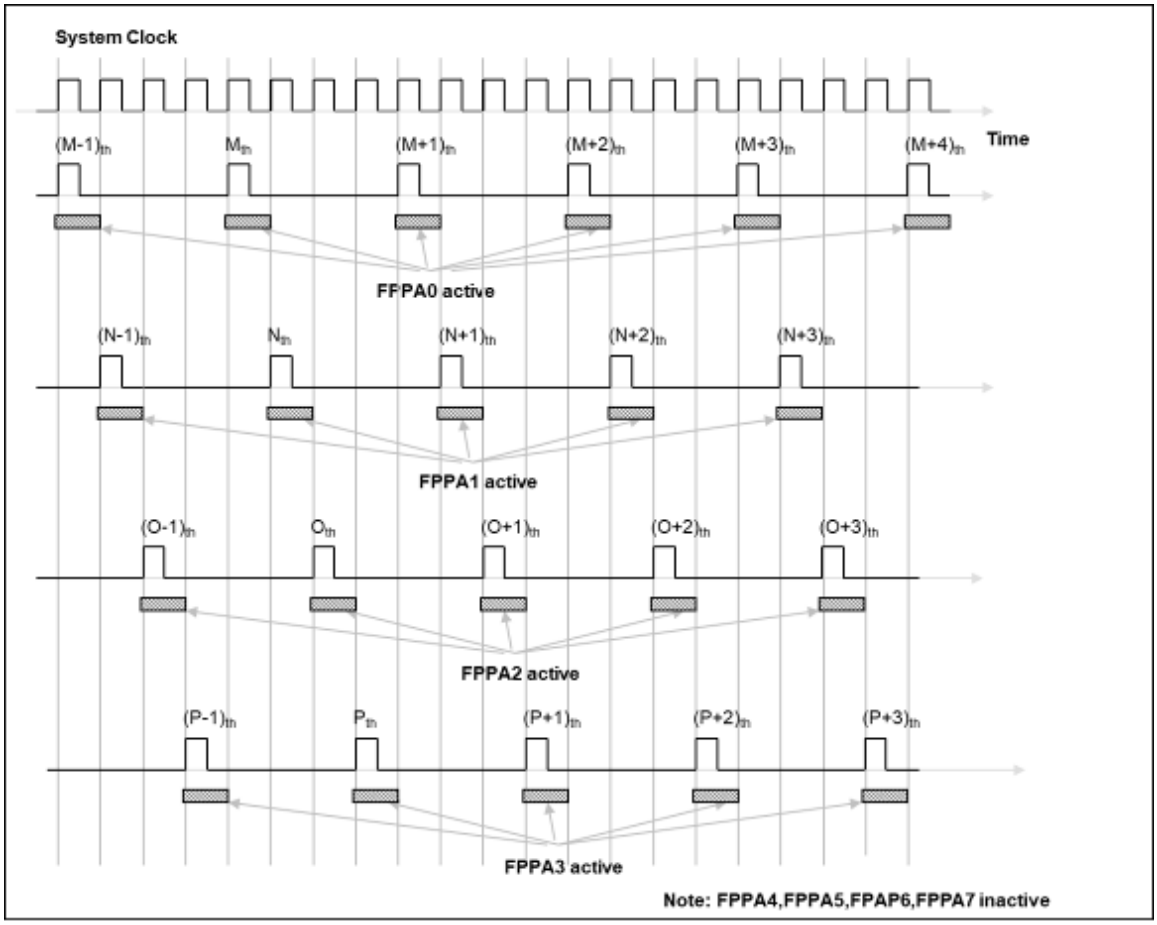


图 3：*pmode*=6 时处理单元时序图

FPPA 单元可以通过允许寄存器编程来启用或停用；上电复位后，只有 FPPA0 是被启用的。系统初始化将从 FPPA0 开始，其余 FPPA 单元可以由用户的程序来决定是否启用。所有 FPPA 单元均可被任意 FPPA 单元停用，包括停用本身这一 FPPA 单元。

5.1.2. 程序计数器

程序计数器（PC）记录下一个执行指令的地址，在每个指令周期后程序计数器会自动递增，以便指令码按顺序从程序存储器取出。某些指令，如分支指令和子程序调用都会改变顺序并放入一个新值到程序计数器。

MF324 程序计数器的位长度是 13。在硬件复位后，FPPA0 的程序计数器为 0、FPPA1 为 1、FPPA2 为 2、FPPA3 为 3、FPPA4 为 4、FPPA5 为 5、FPPA6 为 6、FPPA7 为 7。当中断发生时，FPPA0 的程序计数器会跳转到 0x10 的中断服务程序处。每个 FPPA 单元都具有各自独立的程序计数器来控制其程序执行顺序。

5.1.3. 程序结构

开机后，FPPA0~FPPA7 的程序开始地址依次是 0x000~0x007。中断服务程序的入口地址是 0x010，而且只有 FPPA0 才能接受中断服务。 MF324 的基本软件结构如图 4 所示，使用了四个 FPPA 单元。四个 FPPA 的处理单元的程序代码是被放置在同一个程序空间。除了初始地址和中断入口地址外，处理单元的程序代码可以放在程序存储器任何位置，并没有在特定的地址。开机后，将首先执行 FPPA0Boot，其中将包括系统初始化和启用其它 FPPA 的单元

```
// Page 1
.romadr    0x00
// Program Begin
goto      FPPA0Boot;
goto      FPPA1Boot;
goto      FPPA2Boot;
goto      FPPA3Boot;
//------Interrpt service Routine----------------
.romadr     0x010
pushaf ;
t0sn       intrq.0;       //PA.7 ISR
goto       ISR_PA7;
//------End of ISR---------
//------ Begin of FPPA0 ----------
FPPA0Boot :
  //--- Initialize FPPA0 SP and so on…
…
FPPA0Loop:
…
goto FPPA0Loop:
//------ End of FPPA0 --------
```

```
// Page 2
//------ Begin of FPPA1 ----------
FPPA1Boot :
//--- Initialize FPPA1 SP and so on…
FPPA1Loop:
…
goto FPPA1Loop:
//--------- End of FPPA1 --------
//------ Begin of FPPA2 ----------
FPPA2Boot :
//--- Initialize FPPA2 SP and so on…
FPPA2Loop:
…
goto FPPA2Loop:
//--------- End of FPPA2 --------
//------ Begin of FPPA3 ----------
FPPA3Boot :
//--- Initialize FPPA3 SP and so on…
FPPA3Loop:
…
goto FPPA3Loop:
//--------- End of FPPA3 --------
```

图 4：程序结构

5.1.4. 算术和逻辑单元

算术和逻辑单元（ALU）是用来作整数算术、逻辑、移位和其它特殊运算的单元。运算的数据来源可以从指令、累加器或 SRAM 数据存储器，计算结果可写入累加器或 SRAM。所有的 FPPA 单元在其相应的操作周期分享 ALU 的使用。

5.2. 存储器

5.2.1. 程序存储器 – ROM

MF324 的程序存储器记忆体是 MTP（可多次编程），用来存放数据（包含：数据、表格和中断入口）和要执行的程序指令。所有 FPPA 单元的用户程序代码都存储在 4KW 程序存储器中，如表 1 所示。

复位之后，FPPA0 的初始地址是 0x000，FPPA1 为 0x001，FPPA2 为 0x002，FPPA3 为 0x003，FPPA4 为 0x004，FPPA5 为 0x005，FPPA6 为 0x006，FPPA7 为 0x007。中断入口在 0x010，只有 FPPA0 能使用中断功能。

MTP 存储器从地址“从 0xBE0 到 0xBFF”供系统使用，从“0x008 到 0x00F”和“ 0x0110xBDF”地址空间是用户的程序空间。地址“0x000 to 0x007”为 FPPA 单元的初始地址。

地址	功能
0x000	FPPA0 起始地址– *goto* 指令
0x001	FPPA1 起始地址– *goto* 指令
0x002	FPPA2 起始地址– *goto* 指令
0x003	FPPA3 起始地址– *goto* 指令
0x004	FPPA4 起始地址– *goto* 指令
0x005	FPPA5 起始地址– *goto* 指令
0x006	FPPA6 起始地址– *goto* 指令
0x007	FPPA7 起始地址– *goto* 指令
0x008	用户程序区
•	•
0x00F	用户程序区
0x010	中断入口地址
0x011	用户程序区
•	•
•	•
0xBDF	用户程序区
0xBE0	系统使用
•	•
0xBFF	系统使用

表 1：MF324 程序存储器结构

两个处理单元工作模式下程序存储器分配例子

为了保证用户编写程序时最大的弹性，所有 FPPA 单元共享用户程序存储器，由编译器自动分配程序空间，如果没有特别需求，用户不需要指定地址。使用两个处理单元工作模式下，程序存储器分配情形如表 2 所示：

地址	功能
0x000	FPPA0 起始地址– goto 指令(*goto* 0x020)
0x001	FPPA1 起始地址– goto 指令(*goto* 0x7A1)
0x002	保留
•	•
0x007	保留
0x008	不使用
•	•
0x00F	不使用
0x010	中断入口地址（只给 FPPA0）
0x01F	ISR 结束（取决于用户程序大小）
0x020	FPPA0 用户程序开始
•	•
•	•
0x7A0	FPPA0 程序结束
0x7A1	FPPA1 程序开始
•	•
•	•
0xF37	FPPA1 程序结束
0xF38	不使用
•	•
•	•
0xBDF	不使用
0xBE0	系统使用
•	•
0xBFF	系统使用

表 2：程序存储器分配案例

5.2.2. 数据存储器 – SRAM

图 5 显示了 MF324 数据存储器的结构以及使用，所有的 SRAM 数据存储器均可以透过 FPPA 单元在 1 个时钟周期内直接读取或写入。存取方式可以是字节或位操作。此外 SRAM 数据存储器还充当间接访问方法的数据指针和所有 FPPA 单元的堆栈记忆体。

每个 FPPA 单元的堆栈记忆体使用是独立互不影响的，并定义在数据存储器中。各 FPPA 单元的堆栈指针通过堆栈指针寄存器各自定义，需要的堆栈深度是由使用者来定。堆栈记忆体的调整可完全灵活安排，可以由用户动态调整。

对于间接存取指令而言，数据存储器用作数据指针来当数据地址，所有的数据存储器都可以当做数据指针，这对于间接存取指令是相当灵活和有用的。MF324 的 192 个字节数据存储器都可以利用间接存取指令来存取。

位寻址只能定义在 RAM 区的 0x00 到 0x3F 空间。

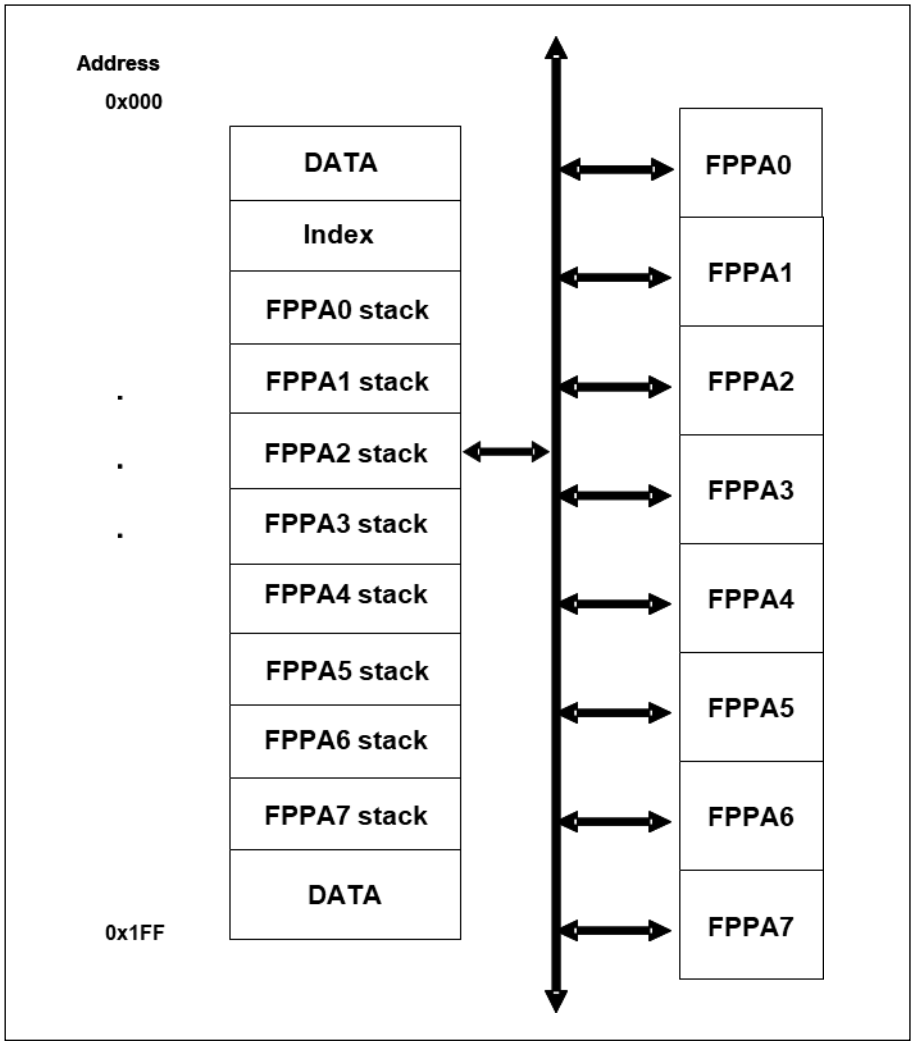


图 5：数据存储器结构和使用

5.2.3. 系统寄存器

MF324 的寄存器地址空间与数据存储空间、MTP 程序空间三者互相独立。

以下是 MF324 的各寄存器存放地址及简要描述：

	+0	+1	+2	+3	+4	+5	+6	+7
0x00	*FLAG*	*FPPAEN*	*SP*	*CLKMD*	*INTEN*	*INTRQ*	*T16M*	
0x08	*INTEN2*	*INTRQ2*			*INTEGS*	*PADIER*		*PA*
0x10	*PAC*	*PAPH*	*PAPL*	*PB*	*PBC*	*GPCC*	*GPCR*	
0x18	*PWMPBC1*	*PWMPBC2*	*PWMGC1*	*PWMGC2*	*PWMGC*	*EARITH*	*ADCRH*	*ADCRL*
0x20	*ADCC*	*ADCM*	*TM4C*	*TM4CTH*	*TM4CTL*	*TM4BH*	*TM4BL*	*TM2C*
0x28	*TM2CT*	*TM2S*	*TM2B*	*TM3C*	*TM3CT*	*TM3S*	*TM3B*	*LCS*
0x30	*ZCPC*	*ZCPS*	*PLSCC*	*PLSCS-*	*MISC*	*M8OP1-*	*M8OP2*	*M8RS1*
0x38	*M8RS0*	*PLSPWH*	*PLSPWL*	*PLSPHH*	*PLSPHL*	*PWMUPBH*	*PWMUPBL*	*PWM0DTH*
0x40	*PWM0DTL*	*PWMADCH*	*PWMADCL*	*PWMDDZVF*	*PWMDDZVR*			*OPR2*

5.2.3.1. ACC 状态标志寄存器 (*FLAG*)，地址= 0x00

位	初始值	读/写	描 述
7 - 4	1111	-	保留。这 4 个位读值为“1”。
3	0	读/写	OV（溢出标志）。当符号操作溢出时，该位被置位。
2	0	读/写	AC（辅助进位标志）。设置该位有种情况： (1) 是进行低半字节加法运算产生进位 (2) 减法运算时，低半字节向高半字节借位
1	0	读/写	C（进位标志）。设置该位有种情况： (1) 加法运算产生进位 (2) 减法运算有借位。进位标志还受带进位标志的 shift 指令影响。
0	0	读/写	Z（零）。当算术或逻辑运算的结果是 0；否则将被清零。

5.2.3.2. 多核使能寄存器 (*MCUEN*), 地址= 0x01

位	初始值	读/写	描 述
7	0	读/写	FPPA7 启用。此位是用来启用 FPPA7。 0/1：停用/启用
6	0	读/写	FPPA6 启用。此位是用来启用 FPPA6。 0/1：停用/启用
5	0	读/写	FPPA5 启用。此位是用来启用 FPPA5。 0/1：停用/启用
4	0	读/写	FPPA4 启用。此位是用来启用 FPPA4。 0/1：停用/启用
3	0	读/写	FPPA3 启用。此位是用来启用 FPPA3。 0/1：停用/启用
2	0	读/写	FPPA2 启用。此位是用来启用 FPPA2。 0/1：停用/启用
1	0	读/写	FPPA1 启用。此位是用来启用 FPPA1。 0/1：停用/启用
0	1	读/写	FPPA0 启用。此位是用来启用 FPPA0。 0/1：停用/启用

5.2.3.3. 选项寄存器 2 (*OPR2*)，地址 = 0x47

位	初始值	读/写	描 述
7 - 5	-	-	保留。请保持为 0
4	-	-	保留。请保持为 0
3	-	-	保留。请保持为 0
2	0	只写	用于选择定时/计数器 16 时钟预分频器。请参见 T16M 寄存器。
1	-	-	保留，请保持为 0
0	0	只写	外部中断 0 引脚选择选项。0 / 1: PA4 / PA3

5.2.3.4. 杂项寄存器 (*MISC*), 地址= 0x34

位	初始值	读/写	描 述
7	0	只写	PWMO0 / PWMO2 / PWMO4，紧急停止 停用/启用
6	0	只写	PWMO1 / PWMO3 / PWMO5，紧急停止 停用/启用
5 - 3	-	-	保留
2	0	只写	停用 LVR 功能： 0 / 1：启用 / 停用
1 - 0	10	只写	看门狗时钟超时时间设定： 00：保留 01：4096 个 ILRC 时钟周期 10：16384 个 ILRC 时钟周期 11：保留

5.3. 堆栈

在每个处理单元的堆栈指针是用来指引堆栈存储器的顶部，该处是用来存储子程序的局部变量和参数的地方；堆栈指针寄存器(*SP*)的地址是 0x02。堆栈指针的位数是 7 位，堆栈存储器是与数据 SRAM 共享，所以堆栈存储器的使用从地址 0x00 开始，不可以超过 128 字节。MF324 每个 FPPA 单元使用的堆栈存储器都可以由用户通过指定堆栈指针寄存器来调整，意味着每个 FPPA 单元的堆栈指针单位深度是可调的，以优化系统性能。下面的示例显示了如何在 ASM 汇编语言下定义堆栈：

```
        .ROMADR0
        GOTO        FPPA0
        GOTO        FPPA1
        ...
        .RAMADR0                    // 地址必需小于 0x100
        WORD        Stack0 [1]      // 1 个 WORD
        WORD        Stack1 [2]      // 2 个 WORD
        ...
FPPA0:
    SP      =       Stack0;         // 指定 Stack0 给 FPPA0 使用,
                                    // 只能有一层呼叫，因为 Stack0[1]
        ...
        call        function1
        ...
FPPA1:
        SP  =       Stack1;         // 指定 Stack1 给 FPPA1 使用,
                                    // 可以有 2 层呼叫，因为 Stack1[2]
        ...
        call        function2
        ...
```

在使用 Mini-C 语言下，由系统软件计算堆栈的深度，使用者不需特别花时间计算，主程序如下：

```
void        FPPA0 (void)
{
            ...
}
```

用户可以在程序分解的窗口里检查堆栈的设定，图 6 表示在 FPPA0 执行前的堆栈状态，系统计算出所需的堆栈空间，并保留该空间给程序使用。

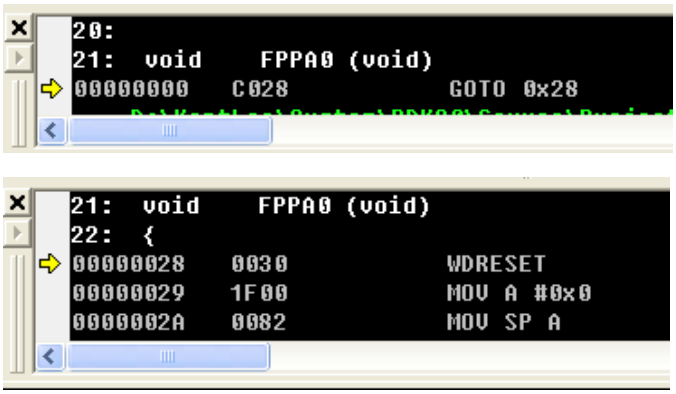


图 6：在 Mini-C 语言下的堆栈设定

5.3.1. 堆栈指针寄存器(*SP*), 地址= 0x02

位	初始值	读/写	描 述
7 - 0	-	读/写	堆栈指针寄存器。读出当前堆栈指针，或写入以改变堆栈指针。

5.4. 程序选项

选项	选择	说明
Security	Enable	MTP 内容加密，程序不允许被读取
	Disable(默认)	MTP 内容不加密，程序可以被读取
LVR	64us	LVR IHRC 消抖 > 64us
	32us	LVR IHRC 消抖 > 32us
	16us	LVR IHRC 消抖 > 16us
	8us	LVR IHRC 消抖 > 8us
	4us	LVR IHRC 消抖 > 4us
	2us	LVR IHRC 消抖 > 2us
	Disable	LVR IHRC 消抖 停用

6. 振荡器和系统时钟

MF324 提供两个振荡电路：内部高 RC 振荡器 (IHRC) 和内部低 RC 振荡器 (ILRC)。这三个振荡器可分别通过寄存器 CLKMD.4 和 CLKMD.2 启用或禁用。

用户可以从这两个振荡器中选择一个作为系统时钟源，并使用 CLKMD 寄存器将所需频率作为系统时钟，以满足不同的应用需求。

振荡器硬件	启用或停用选择	开机后默认状态
IHRC	*CLKMD*.4	启用
ILRC	*CLKMD*.2	启用

表 3：MF324 提供 2 个振荡器电路

6.1. 内部高频振荡器和内部低频振荡器

启动后，启用 IHRC 和 ILRC 振荡器。IHRC 的频率可通过 IHRCR 寄存器进行校准，以消除工艺变化；通常校准为 16MHz。校准后的频率偏差通常在 2%以内，但仍会随着电源电压和工作温度的变化而轻微漂移。

ILRC 的频率也会随工艺、电源电压和温度的变化而变化。有关 IHRC / ILRC 频率与 VDD 的关系以及 IHRC / ILRC 频率与温度的关系，请参见测量图表。

MF324 烧录工具提供 IHRC 频率校准（通常高达 16MHz），以消除工厂生产造成的频率漂移。ILRC 没有校准操作。对于需要精确定时的应用，请勿将 ILRC 时钟用作参考时间。

注意：如果 PWMGC 与源自 IHRC 的时钟源配合使用，软件必须在禁用 IHRC 之前先禁用 PWMGC。

©Copyright 2026, PADAUK Technology Co. Ltd

6.2. 系统时钟和 IHRC 校准

6.2.1. 系统时钟

系统时钟的时钟源来自 IHRC 和 ILRC，MF324 中系统时钟的硬件图如图 7 所示。

上电后，运行的系统时钟为 ILRC/1，用户可以通过设置 CLKMD 寄存器随时更改系统时钟，写入 CLKMD 寄存器后，新的系统时钟将立即转换为新的时钟。

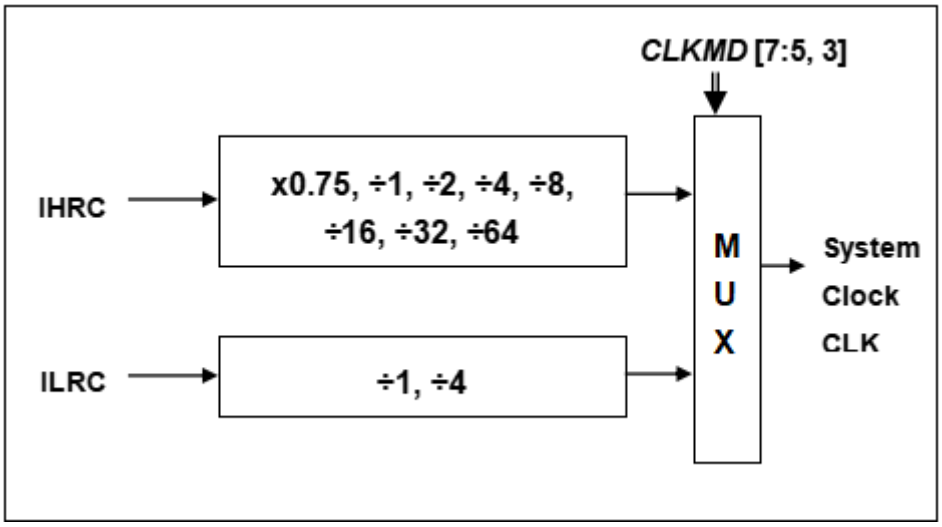


图 7：晶体振荡的连接

6.2.1.1. 时钟模式寄存器(CLKMD)，地址= 0x03

位	初始值	读/写	描 述	
7-5	111	读写	系统时钟选择	
			Type 0, *CLKMD*[3]=0	Type 1, *CLKMD*[3]=1
			000：IHRC/4 001：IHRC/2 010 / 011 / 100 / 101：保留 110：ILRC/4 111：ILRC (默认)	000：IHRC/16 001：IHRC/8 010：保留 011：IHRC/32 100：IHRC/64 101 / 11x：保留
4	1	读写	启用 IHRC 振荡器。0 / 1：停用/启用	
3	0	读写	时钟类型选择。该位用于选择位 [7:5] 中的时钟类型 0 / 1：类型 0 / 类型 1	
2	1	读写	启用 ILRC。0 / 1：禁用/启用 如果禁用 ILRC，看门狗定时器也将被禁用。	
1	1	读写	启用看门狗。0 / 1：禁用/启用	
0	0	读写	引脚 PA5/PRSTB 功能。0 / 1：PA5 / PRSTB	

6.2.2. 频率校准

在芯片生产制造时，IHRC 频率和 Bandgap 参考电压都有可能稍微不同，MF324 提供 IHRC 频率 Bandgap 校准来消除这些差异，校准功能可以被用户的程序选择并编译，同时这个命令会自动嵌入用户的程序里面。

校准命令如下所示：

.ADJUST_IC SYSCLK=IHRC/(**p1**), IHRC=(**p2**)MHz, VOUT_5V=(**p3**)V, Bandgap=(**p4**);

式中，**p1**=2, 4, 8, 16, 32, 64; 以提供不同的系统时钟。

p2=16 ~ 18; 校准芯片到不同的频率，通常选择 16MHz。

p3=2.7 ~ 5.5; 根据不同的电源电压校准芯片。

p4= 打开或者关闭 Bandgap 校准。

通常情况下，ADJUST_IC 是开机后的第一个命令，用以设定系统的工作频率。IHRC 频率校准的程序只在将程序代码写入 MTP 存储器的时候执行一次，此后不会再被执行。

如果 IHRC 校准选择不同的选项，开机后的系统状态也是不同的。IHRC 频率校准以及系统时钟的选项，如表 4 所示：

SYSCLK	CLKMD	IHRCR	Description
○ Set IHRC / 2	= 34h (IHRC / 2)	有校准	IHRC 校准到 16MHz,CLK=8MHz (IHRC/2)
○ Set IHRC / 4	= 14h (IHRC / 4)	有校准	IHRC 校准到 16MHz, CLK=4MHz (IHRC/4)
○ Set IHRC / 8	= 3Ch (IHRC / 8)	有校准	IHRC 校准到 16MHz,CLK=2MHz (IHRC/8)
○ Set IHRC / 16	= 1Ch (IHRC / 16)	有校准	IHRC 校准到 16MHz, CLK=1MHz (IHRC/16)
○ Set IHRC / 32	= 7Ch (IHRC / 32)	有校准	IHRC 校准到 16MHz,CLK=0.5MHz (IHRC/32)
○ Set ILRC	= E4h (ILRC / 1)	有校准	IHRC 校准到 16MHz,CLK=ILRC
○ Disable	不改变	不改变	IHRC 不校准，CLK 不改变

表 4：IHRC 频率校准选项

下面显示在不同的选项下，MF324 不同的状态：

(1) ***.ADJUST_IC SYSCLK=IHRC/4, IHRC=16MHz, V_{DD}=5V***

开机后，CLKMD = 0x14：

a. IHRC 的校准频率为 16MHz@V_{DD}=5V，启用 IHRC 的硬件模块
b. 系统时钟 = IHRC/4 = 4MHz
c. 看门狗被停用，启用 ILRC，PA5 是在输入模式

(2) **.ADJUST_IC SYSCLK=ILRC, IHRC=16MHz, *V_{DD}=5V***

开机后，CLKMD = 0xE4：

a. IHRC 的校准频率为 16MHz@V_{DD}=5V，停用 IHRC 的硬件模块
b. 系统时钟 = ILRC
c. 看门狗被停用，启用 ILRC，PA5 是在输入模式

(3) ***.ADJUST_IC DISABLE***

开机后，CLKMD 寄存器没有改变（没任何动作）：

a. IHRC 不校准并且 IHRC 模块停用
b. 系统时钟 = ILRC
c. 看门狗被启用，启用 ILRC，PA5 是在输入模式

6.2.2.1. 特别声明

(1) IHRC 的校正操作是在 IC 烧录时进行的。

(2) IC 塑封材料（不论是封装用还是 COB 用的黑胶）的特性会对 IHRC 的频率有一定影响。如果用户在 IC 盖上塑封材料前进行烧录，然后再封上塑封材料，则可能造成 IHRC 的特性偏移超出规格的现象，正常情况下频率会变慢一些。

(3) 上述问题通常发生在用户使用 COB 封装或是委托我司进行晶圆代烧(QTP)时。此情况下我司将不对频率超出规格的情况负责。

(4) 用户可按自身经验进行一些补偿性调整，例如把 IHRC 的目标频率调高 0.5%-1%左右，令封装后 IC 的 IHRC 频率更接近目标值。

6.2.3. 系统时钟切换

IHRC 校准后，透过 CLKMD 寄存器的设定，MF324 系统时钟可以随意在 IHRC 和 ILRC、EOSC 之间切换。**但必须注意，不可在切换系统时钟的同时把原时钟源关闭。**例如：从 A 时钟源切换到 B 时钟源时，应该先把系统时钟源切换到 B，然后再关闭 A 时钟源。请参阅 IDE：“帮忙” → “使用手册” → “IC 介绍” → “缓存器介绍” → “CLKMD”。

例 1: 系统时钟从 ILRC 切换到 IHRC/4

```
...                          //  系统时钟为 ILRC
CLKMD.4     =   1;           //  先打开 IHRC，可以提高抗干扰能力
CLKMD       =   0x14;        //  切换为 IHRC/4，ILRC 不能在这里停用
// CLKMD.2  =   0;           //  假如需要，ILRC 可以在这里停用
...
```

例 3: 系统时钟从 IHRC/8 切换到 IHRC/4

```
...                          //  系统时钟为 IHRC/8，ILRC 为启用
CLKMD       =   0x14;        //  切换为 IHRC/4
...
```

例 4: 系统可能当机，如果同时切换时钟和关闭原来的振荡器

```
...                          //  系统时钟为 ILRC
CLKMD       =   0x10;        //  不能从 ILRC 切换到 IHRC/4，同时又关闭 ILRC 振荡器
...
```

7. 复位

导致 MF324 复位的原因有很多，主要有四个：上电复位、LVR 复位、看门狗超时溢出复位和 PRSTB 引脚复位。复位后，系统将重新启动。程序计数器将跳转到地址 0x000，MF324 中的大部分寄存器将被设置为默认值。

7.1. 上电复位 - POR

开机时，POR（上电复位）是用于复位 MF324；但是，上电后电源电压可能不太稳定，为确保单片机是工作在电压稳定的状态，在执行第一条指令之前，会等待 2547 个 ILRC 时钟周期，这时间就是 T_{SBP}，如图 8 所示。开机后，默认的系统时钟是 ILRC。如果用户希望将系统时钟源从 ILRC 切换到 IHRC，则必须启用相应的振荡器模块，并确保时钟已经稳定。

发生上电复位时，MF324 数据存储器的值处于不确定的状态。

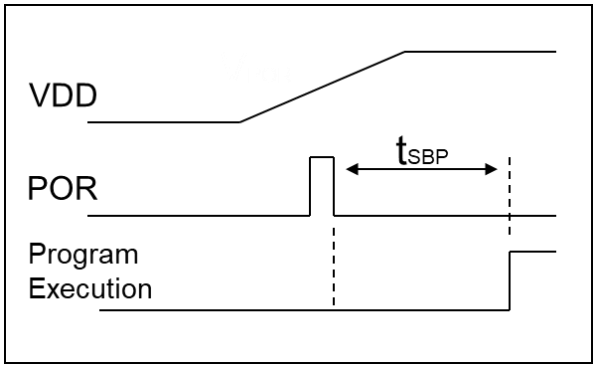


图 8：上电复位时序图

图 9 显示的是典型开机流程。请注意，上电复位后 FPPA1~FPPA7 是停用，建议不要在 FPPA0 以及系统初始化完成前，启用 FPPA1~FPPA7。

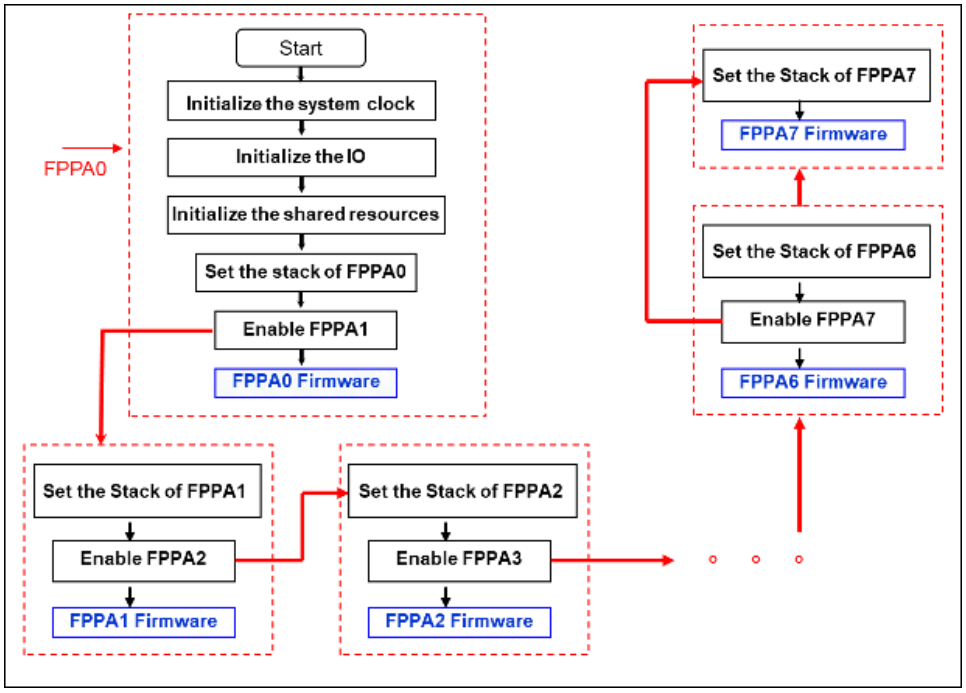


图 9：开机流程

7.2. 低压复位- LVR

用户可以根据自己的需求选择不同的操作系统时钟；所选的操作系统时钟应与电源电压和 LVR 电平相结合，以确保系统稳定。如果 VDD 下降到 LVR 电平以下，系统将发生 LVR 复位，其时序图如图 10 所示。

LVR 复位后，当 VDD>VDR （SRAM 数据保持电压）时，SRAM 数据将被保留。但是，如果再次上电后清零 SRAM，数据将无法保留，当 VDD<VDR 时，数据存储器处于不确定状态。

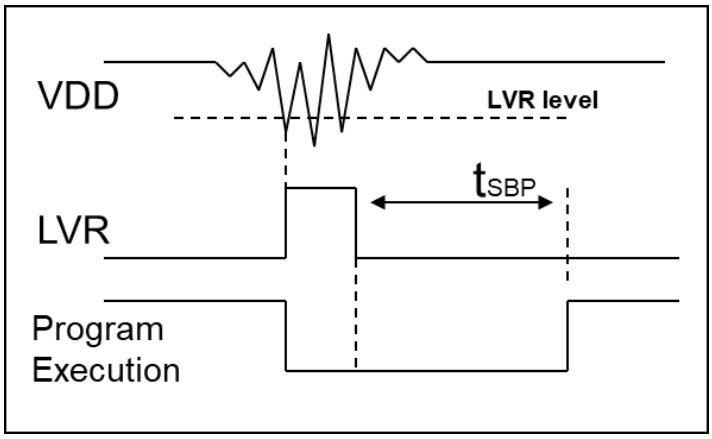


图 10：低压复位时序图

使用者可以在不同的需求下选择不同的系统时钟，选定的系统时钟应与电源电压和 LVR 的基准位结合起来才能使系统稳定。LVR 的基准位是在编译过程中选择，不同系统时钟对应的 LVR 设定，请参考章节 13.1 中系统时钟的最低工作电压。

下面是工作频率、电源电压和 LVR 水平设定的建议：

SYSCLK	VDD	LVR
8MHz	≧2.75V	2.75V
4MHz	≧ 2.0V	2.0V

表 5：LVR 设置参考

用户可将 MISC.2 设置为 1 以禁用 LVR 功能。此时应确保 VDD 电压高于芯片的最低工作电压，否则 IC 可能无法正常工作。

杂项寄存器 (*MISC*), 地址= 0x34			
位	初始值	读/写	描 述
7	0	只写	PWMO0 / PWMO2 / PWMO4 紧急停止 停用/ 启用
6	0	只写	PWMO1 / PWMO3 / PWMO5 紧急停止 停用/ 启用
5 - 3	-	-	保留
2	0	只写	停用 LVR 功能 0 / 1：启用 / 停用
1 - 0	00	只写	看门狗时钟超时时间设定： 00：保留 01：4096 个 ILRC 时钟周期 10：16384 个 ILRC 时钟周期 11：保留

7.3. 看门狗超时溢出复位

看门狗(WDT)是一个计数器，其时钟源来自 ILRC，看门狗默认为开，但程序执行 ADJUST_IC 时，会将看门狗关闭，若要使用看门狗，需重新配置打开。当 ILRC 关闭时，看门狗也会失效。ILRC 的频率有可能因为工厂制造的变化，电源电压和工作温度而漂移很多，使用者必须预留安全操作范围。

另外，在复位或唤醒事件后，看门狗的周期也会比预期的短。建议在这些事件之后通过 *wdreset* 指令清除 WDT，以确保在 WDT 超时之前有足够的时钟周期。

为确保看门狗在超时溢出之前被清零，在安全时间内，可以用指令 *wdreset* 清零看门狗。在上电复位(POR)或任何时候使用 *wdreset* 指令，看门狗都会被清零。

当 WDT 超时时，MF324 将被复位，重新开始执行程序。看门狗定时器的相对时序图如图 11 所示。

发生看门狗重置时，MF324 的数据存储器将被保留。

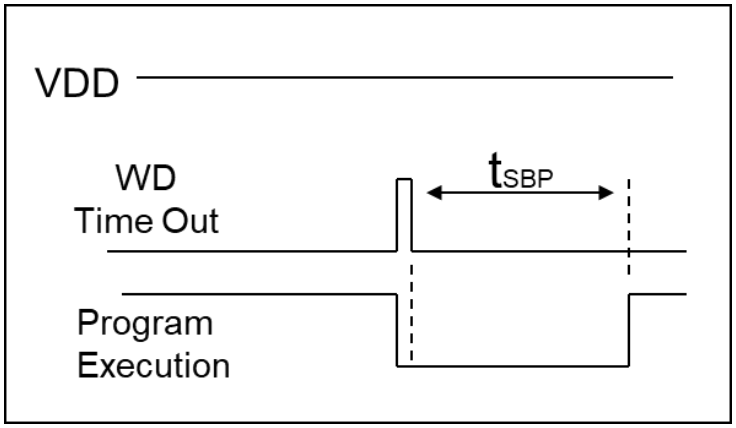


图 11：看门狗超时溢出的相关时序

利用寄存器 *MISC*[1:0]可选择两种不同的看门狗超时时间，利用 *CLKMD*.1 可以选择将看门狗功能停用。

时钟控制寄存器(CLKMD)，地址 = 0x03			
位	初始值	读/写	描 述
7 – 5	111	读/写	系统时钟选择
4	1	读/写	内部高频 RC 振荡器功能。 0/1：停用/启用
3	0	读/写	时钟类型选择
2	**1**	**读/写**	**内部低频 RC 振荡器功能。 0/1：停用/启用** **当 ILRC 关闭时，看门狗也会失效**
1	**1**	**读/写**	**看门狗功能。 0/1：停用/启用**
0	**0**	**读/写**	**引脚 PA5/PRSTB 功能。0 / 1：PA5 / PRSTB**

7.4. 外部复位 - PRSTB

MF324 支持外部复位功能，其外部复位引脚与 PA5 共享同一个 IO 端口。使用外部复位功能需要：

(1) 设定 PA5 为输入；

(2) 设定 *CLKMD*.0=1，使 PA5 为外部 PRSTB 输入脚位。

在外部复位引脚为高电平时，系统处于正常工作状态；一旦复位引脚检测到低电平，系统即发生复位。PRSTB 复位时序图如图 12 所示。

当发生 PRSTB 复位时，MF324 数据存储器的值将被保留。

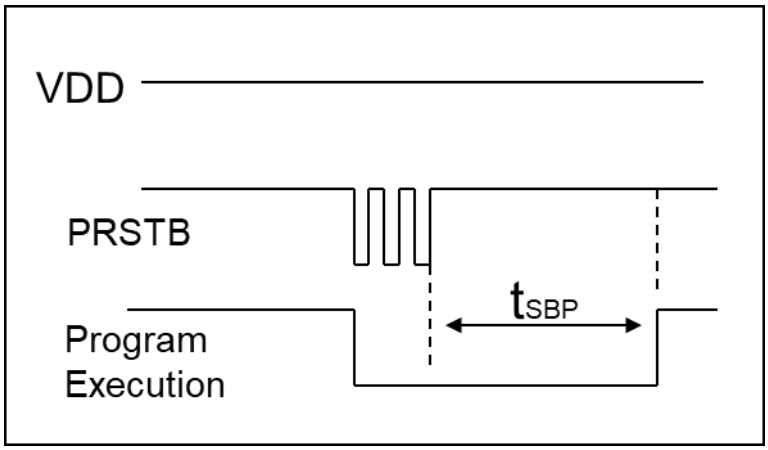


图 12：外部引脚复位的相关时序

8. 中断

MF324 有 9 个中断源：

1. 三个外部中断引脚（PA3、PA4、PA6，类型由 INTEGS 寄存器指定）
2. GPC 中断（类型由 INTEGS 寄存器指定）
3. Timer16 中断（类型由 INTEGS 寄存器指定）
4. PWM 中断
5. ADC 中断
6. Timer2 中断
7. LC 中断
8. Timer3 中断
9. Timer4 中断

每个中断请求源都有自己的中断控制位启用或停用它。中断硬件框图请参考图 13。所有的中断请求标志位是由硬件置位并且并通过软件写寄存器 *INTRQ/INTRQ2* 清零。中断请求标志设置点可以是上升沿或下降沿或两者兼而有之，这取决于对寄存器 *INTEGS/INTEGS2* 的设置。所有的中断请求源最后都需由 *engint* 指令控制（启用全局中断）使中断运行，以及使用 *disgint* 指令（停用全局中断）停用它。

只有 FPPA0 可以接受中断请求，其他 FPPA 单元不被中断影响。中断堆栈是共享数据存储器，其地址由堆栈寄存器 SP 指定。由于程序计数器是 15 位宽度，堆栈寄存器 SP 位 0 应保持 0。此外，用户可使用 *pushaf/popaf* 指令将 ACC 寄存器和标志寄存器的值存储至/从栈存储器中恢复。

由于堆栈与数据存储器共享，用户应该谨慎使用存储器。可由软件编程调整栈点在存储器里的位置，每个 FPPA 单元堆栈指针的深度可以完全由用户指定，以实现最大的系统弹性。

在中断服务程序中，可以通过读取寄存器 *INTRQ/INTRQ2* 知道中断发生源。

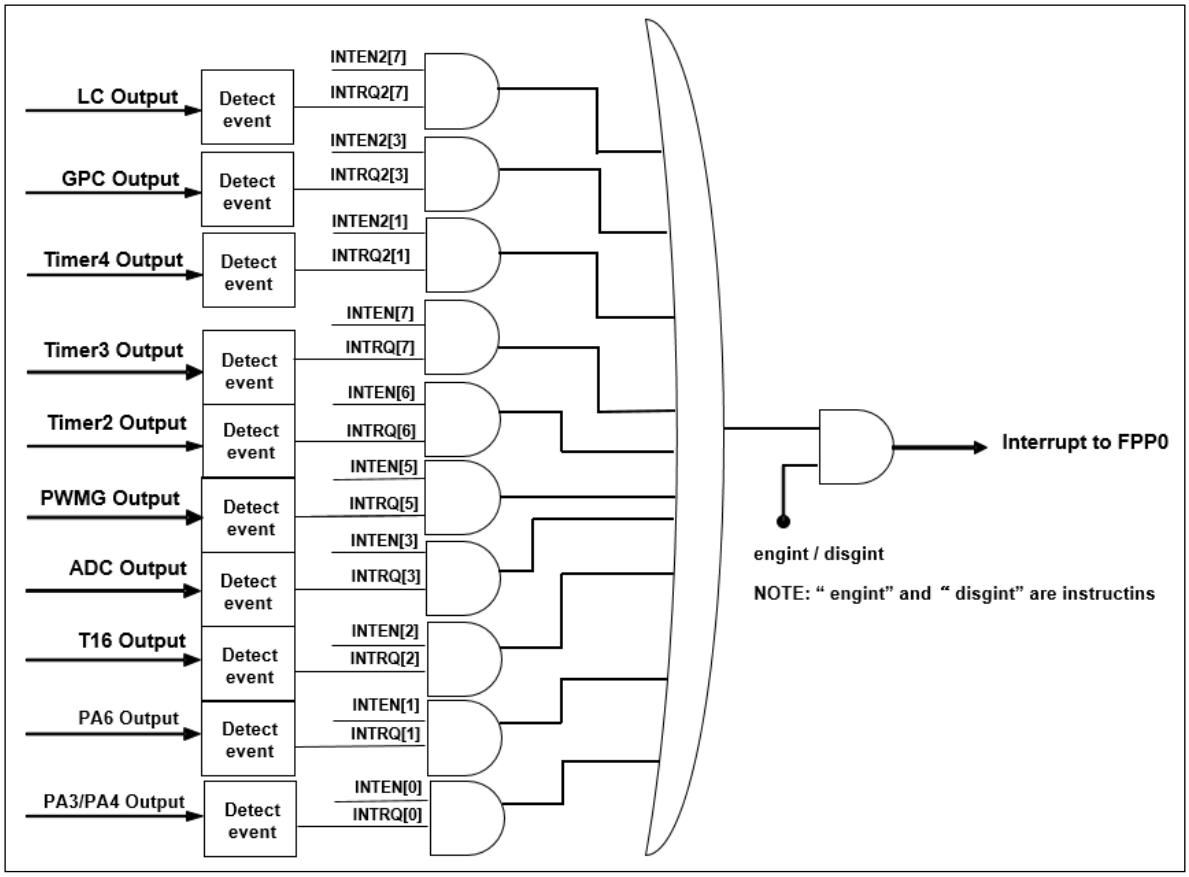


图 13：中断控制硬件图

选项寄存器 2 (*OPR2*)，地址= 0x47			
位	初始值	读/写	描 述
7 – 5	-	-	保留。请保持为 0.
4	-	-	保留。请保持为 0.
3	-	-	保留。请保持为 0.
2	0	只写	用于选择定时/计数器 16 时钟预分频器。请参见 T16M 寄存器
1	-	-	保留。请保持为 0.
0	**0**	只写	**外部中断 0 引脚选择选项。0 / 1: PA4 / PA3**

8.1. 中断允许寄存器 (*INTEN*)，地址= 0x04

位	初始值	读/写	描 述	
7	0	读/写	启用从 Timer3 的溢出中断。0/1：停用/启用	
6	0	读/写	启用从 Timer2 的溢出中断。0/1：停用/启用	
5	0	读/写	启用从 PWM 产生器的溢出中断。0/1：停用/启用	
4	-	-	保留。请保持为 0。	
3	0	读/写	启用从 ADC 的中断。0/1：停用/启用	
2	0	读/写	启用从 Timer16 的溢出中断。0/1：停用/启用	
1	0	读/写	启用从 PA6 的溢出中断。0/1：停用/启用	
0	0	读/写	*OPR*2.1=0	启用从 PA4 的溢出中断。0/1：停用/启用
			*OPR*2.1=1	启用从 PA3 的溢出中断。0/1：停用/启用
其中，OPR2 是选择寄存器，地址=0x47。				

8.2. 中断允许 2 寄存器 (*INTEN2*)，地址= 0x08

位	初始值	读/写	描 述
7	0	-	启用 LC 中断。0 / 1：停用/启用
6	-	-	保留。请保持为 0。
5	-	-	保留。请保持为 0。
4	-	-	保留。请保持为 0。
3	0	读/写	启用从 GPC 的中断。0/1：停用/启用
2	-	-	保留。请保持为 0。
1	0	读/写	启用从 Timer4 的中断。0/1：停用/启用
0	-	-	保留。请保持为 0。

8.3. 中断请求寄存器(*INTRQ*)，地址= 0x05

位	初始值	读/写	描 述	
7	-	读/写	Timer3 的中断请求，此位是由硬件置位并由软件清零。 0/1：不请求/请求	
6	-	读/写	Timer2 的中断请求，此位是由硬件置位并由软件清零。 0/1：不请求/请求	
5	-	读/写	PWM 产生器的中断请求，此位是由硬件置位并由软件清零。 0/1：不请求/请求	
4	-	-	保留。请保持为 0。	
3	-	读/写	ADC 的中断请求，此位是由硬件置位并由软件清零。 0/1：不请求/请求	
2	-	读/写	Timer16 的中断请求，此位是由硬件置位并由软件清零。 0/1：不要求/请求	
1	-	读/写	PA6 的中断请求，此位是由硬件置位并由软件清零。 0/1：不请求/请求	
0	-	读/写	*OPR2*.0=0	PA4 的中断请求，此位是由硬件置位并由软件清零。 0/1：不请求/请求
			OPR2.0=1	PA3 的中断请求，此位是由硬件置位并由软件清零。 0/1：不请求/请求
其中，OPR2 是可选寄存器，地址= 0x47。				

8.4. 中断请求 2 寄存器(INTRQ2)，地址= 0x09

位	初始值	读/写	描 述
7	-	读/写	LC 的中断请求，该位由硬件设置，由软件清除 0 / 1：无请求/有请求
6	-	-	保留。请保持为 0。
5	-	-	保留。请保持为 0。
4	-	-	保留。请保持为 0。
3	-	读/写	GPC 的中断请求，此位是由硬件置位并由软件清零。0/1：不请求/请求
2	-	-	保留。请保持为 0。
1	-	读/写	Timer4 的中断请求，此位是由硬件置位并由软件清零。0/1：不请求/请求
0	-	-	保留。请保持为 0。

8.5. 中断缘选择寄存器 (INTEGS), 地址= 0x0C

位	初始值	读/写	描 述
7	0	读/写	PWM 中断选择 0：PWM 计数器边界模式 1：PWM 计数器零模式
6-5	00	读/写	GPC 中断边沿选择 00：上升沿和下降沿均触发中断 01：上升沿触发中断 10：下降沿触发中断 11：保留
4	0	读/写	Timer16 中断缘选择 0：上升缘请求中断 1：下降缘请求中断
3 - 2	00	读/写	PA6 中断缘选择 00：上升缘和下降缘都请求中断 01：上升缘请求中断 10：下降缘请求中断 11：保留
1 - 0	00	读/写	PA3/PA4 中断缘选择 00：上升缘和下降缘都请求中断 01：上升缘请求中断 10：下降缘请求中断 11：保留 注：如果 OPR2.0=0，则选择 PA4；如果 OPR2.0=1，则选择 PA3

注意：

INTEN/INTEN2，*INTRQ/INTRQ2* 没有初始值，所以要使用中断前，一定要根据需要设定数值。即使 *INTEN/INTEN2* 为 0，*INTRQ/INTRQ2* 还是会被中断发生源触发。

8.6. 中断工作流程

一旦发生中断，其具体工作流程如下：

(1) 程序计数器将自动存储到 *SP* 寄存器指定的堆栈存储器。

(2) 新的 *SP* 将被更新为 SP+2。

(3) 全局中断将自动被停用。

(4) 将从地址 0x010 获取下一条指令。

中断完成后，发出 *reti* 指令返回既有的程序，其具体工作流程如下：

(1) 从 *SP* 寄存器指定的堆栈存储器自动恢复程序计数器。

(2) 新的 *SP* 将被更新为 SP-2。

(3) 全局中断将自动启用。

(4) 下一条指令将是中断前原来的指令。

8.7. 中断的一般步骤

当使用中断函数时，程序应该是:

步骤 1：设定 *INTEN / INTEN2* 寄存器，开启需要的中断的控制位。

步骤 2：清除 *INTRQ* / INTRQ2 寄存器。

步骤 3：主程序中，使用 *engint* 指令（启用全局中断）允许 CPU 的中断功能。

步骤 4：等待中断。中断发生后，跳入中断子程序。

步骤 5：当中断子程序执行完毕，返回主程序。

跳入中断子程序处理时，可使用 *pushaf* 指令来保存 *ALU* 和 *FLAG* 寄存器数据，并在 *reti* 之前，使用 *popaf* 指令复原。一般步骤如下：

```
void Interrupt (void)    // 中断发生后，跳入中断子程序
{                        // 自动进入 disgint 的状态，CPU 不会再接受中断
        PUSHAF;
        …
        POPAF;
}   // 系统自动填入 reti，直到执行 reti 完毕才自动恢复到 engint 的状态
```

* 在主程序中，可使用 *disgint* 指令关闭所有中断

8.8. 使用中断举例

使用者必须预留足够的堆栈存储器以保存中断向量，一级中断需要两个字节，两级中断需要四个字节。下面的示例程序演示了如何处理中断，请注意，处理中断和 *pushaf* 是需要四个字节堆栈存储器。

```
 void          FPPA0     (void)
 {
     ...
     $   INTEN   PA4;        // INTEN =1; 当 PA4 准位改变，产生中断请求
     INTRQ   =   0;          // 清除 INTRQ
     INTRQ2   =   0;         // 清除  INTRQ2

     ENGINT                  // 启用全局中断
     ...
     DISGINT                 // 停用全局中断
     ...
 }
void Interrupt   (void)      // 中断程序
{
     PUSHAF                  // 存储 ALU 和 FLAG 寄存器

     //   如果  INTEN.PA4 在主程序会动态开和关，则表达式中可以判断 INTEN.PA0 是否为 1。
     //   例如：  If (INTEN.PA4 && INTRQ.PA4)   {…}

     //   如果 INTEN.PA4 一直在使能状态
     //   用户可以省略  INTEN.PA4  判断，以加快中断服务例程的速度。

     If (INTRQ.PA4)
     {                           // 此处为  PA4  中断服务例程
          INTRQ.PA4 = 0;         // 删除相应位（以  PA4  为例）
          ...
     }
     ...
  // (X:) INTRQ = 0;            // 不建议在中断程序最后，才使用  INTRQ = 0 一次全部清除
                                // 因为它可能会把刚发生而尚未处理的中断，意外清除掉.
    POPAF                       // 恢复 ALU 和 FLAG 寄存器
}
```

9. I/O 端口

9.1. IO 相关寄存器

9.1.1. 端口 A 数字输入启用寄存器(PADIER)，地址= 0x0D

位	初始值	读/写	描 述
7-0	0xFF	只写	启用 PA7~PA0 数字输入 1 / 0：启用 / 停用

9.1.2. 端口 A 数据寄存器(PA)，地址 = 0x0F

位	初始值	读/写	描 述
7 - 0	0x00	读/写	端口 A 数据寄存器

9.1.3. 端口 A 控制寄存器(PAC)，地址 = 0x10

位	初始值	读/写	描 述
7 - 0	0x00	读/写	端口 A 控制寄存器。这些寄存器是用来定义端口 A 每个相应的引脚的输入模式或输出模式。0 / 1：输入/输出 注意：PA5 仅支持输入模式

9.1.4. 端口 A 上拉控制寄存器(PAPH)，地址= 0x11

位	初始值	读/写	描 述
7 - 0	0x00	读/写	端口 A 上拉寄存器。该寄存器用于在端口 A 的每个相应引脚上启用内部上拉装置。 0 / 1：停用/启用

9.1.5. 端口 A 下拉控制寄存器 (PAPL)，地址= 0x12

位	初始值	读/写	描 述
7 - 0	0x00	读/写	端口 A 下拉寄存器。该寄存器用于在端口 A 的每个相应引脚上启用内部下拉装置。 0 / 1：停用/启用 注意：PA5 仅为输入模式，不支持下拉功能

9.1.6. 端口 B 数据寄存器 (PB)，地址= 0x13

位	初始值	读/写	描 述
7 - 0	0x00	读/写	端口 B 数据寄存器

9.1.7. 端口 B 控制寄存器(PBC)，地址= 0x14

位	初始值	读/写	描 述
7 - 0	0x00	读/写	端口 B 控制寄存器。该寄存器用于定义端口 B 各对应引脚的输出模式（注：仅支持输出模式配置）。0 / 1：无功能 / 输出模式

©Copyright 2026, PADAUK Technology Co. Ltd

9.2. IO 结构及功能

9.2.1. IO 引脚的结构

MF324 的所有 IO 引脚都具有相同的结构，如下图 14。

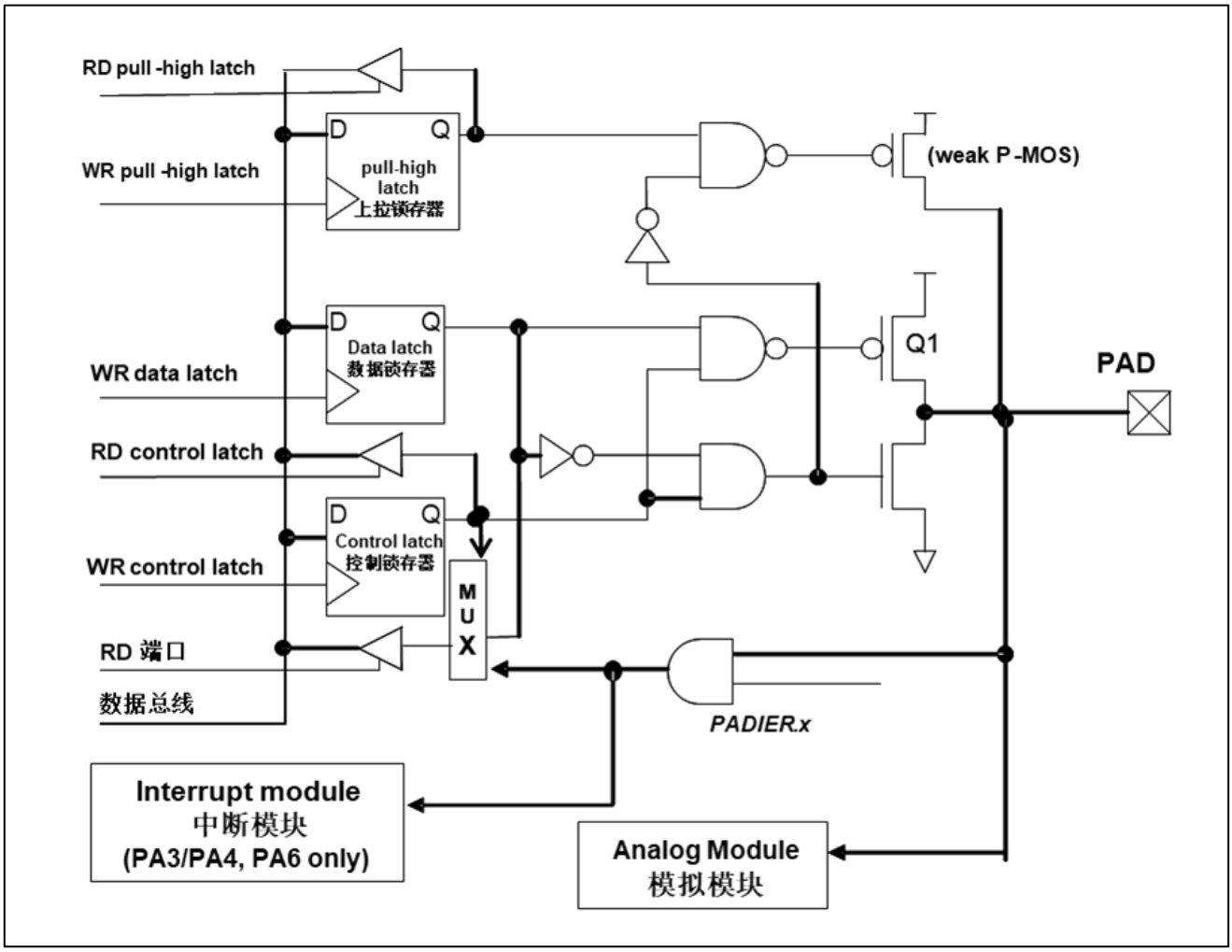


图 14：引脚缓冲区硬件图

9.2.2. IO 引脚的一般功能

(1) **输入、输出功能：**

MF324 除端口 B 及 PA5 之外的所有 IO 引脚都可编程设定为数字输入或模拟输入、输出低或输出高。

通过配置数据寄存器 (PA)、控制寄存器 (PAC)、拉高寄存器 (PAPH) 和拉低寄存器 (PAPL)，可以将每个 IO 引脚独立设置为不同的状态。

对于选择模拟功能的引脚，寄存器 PxDIER 中的相应位应设置为低电平，以防止漏电流。当设置为输出低电平时，拉高电阻会自动关闭。

如果用户要读取引脚状态，请注意在读取数据端口之前应将其设置为输入模式。如果用户在数据端口设置为输出模式时读取数据，读取的数据将来自数据寄存器，而不是 IO 焊盘。

例如，表 6 显示了端口 A 第 0 位的配置表。

PA.0	*PAC.0*	*PAPH.0*	描述
X	0	0	输入，没有弱上拉电阻
X	0	1	输入，有弱上拉电阻
0	1	X	输出低电位，没有弱上拉电阻
1	1	0	输出高电位，没有弱上拉电阻
1	1	1	输出高电位，有弱上拉电阻

表 6：PA0 设定配置表

通过配置数据寄存器（PBC），数字输出端口 B 可独立配置为不同状态。

(2) 外部中断功能：

当 IO 作为外部中断引脚时，PxDIER 相应位应设置高。例，当 PA0 用来作为外部中断引脚时，PADIER.0 应设置高。

9.2.3. IO 的使用与设定

(1) IO 作为数字输入

◆ 将 IO 设为数字输入时，Vih 与 Vil 的准位，会随着工作电压和温度有变动。请参考 Vih 最小值和 Vil 最大值。

◆ 内部上拉电阻值也将随着电压、温度与引脚电压而变动，并非为固定值。

(2) IO 作为数字输入和打开唤醒功能

◆ 用 PxC 寄存器，将 IO 设为输入。

◆ 用 PxDIER 寄存器，将对应的位设为 1 以启用数字输入。

◆ 为了防止 PA 中没有用到的 IO 口漏电，PADIER[1:2]需要常设为 0。

(3) PA5 作为 PRSTB 输入

◆ 设定 PA5 为输入。

◆ 设定 CLKMD.0=1，使 PA5 为外部 PRSTB 输入脚位。

10. Timer / PWM 计数器

10.1. 16 位计数器 (Timer16)

10.1.1. Timer16 介绍

MF324 内置一个 16 位硬件计数器 Timer16，其模块框图如图 15。

计数器时钟源由寄存器 *T16M*[7:5]来选择，在时钟送到 16 位计数器(counter16)之前，*T16M*[4:3]和 *OPR2*[2]可对时钟进行预分频处理，有÷1、÷2、÷4、÷8、÷16、÷32、÷64、÷128 等八种选项，让计数范围更大。

注意：系统时钟不能慢于 T16 时钟。

T16M[2:0]用于选择 Timer16 的中断源，其来自于 16 位计数器的位 8 到位 15。当计数器溢出时，Timer16 就触发中断。经由寄存器 *INTEGS*.4，可选择中断类型是上升沿触发或下降沿触发。

16 位计数器仅执行向上计数操作，可通过发布 stt16 指令从数据存储器中存储计数器初始值，并通过发布 ldt16 指令将计数值加载到数据存储器中。

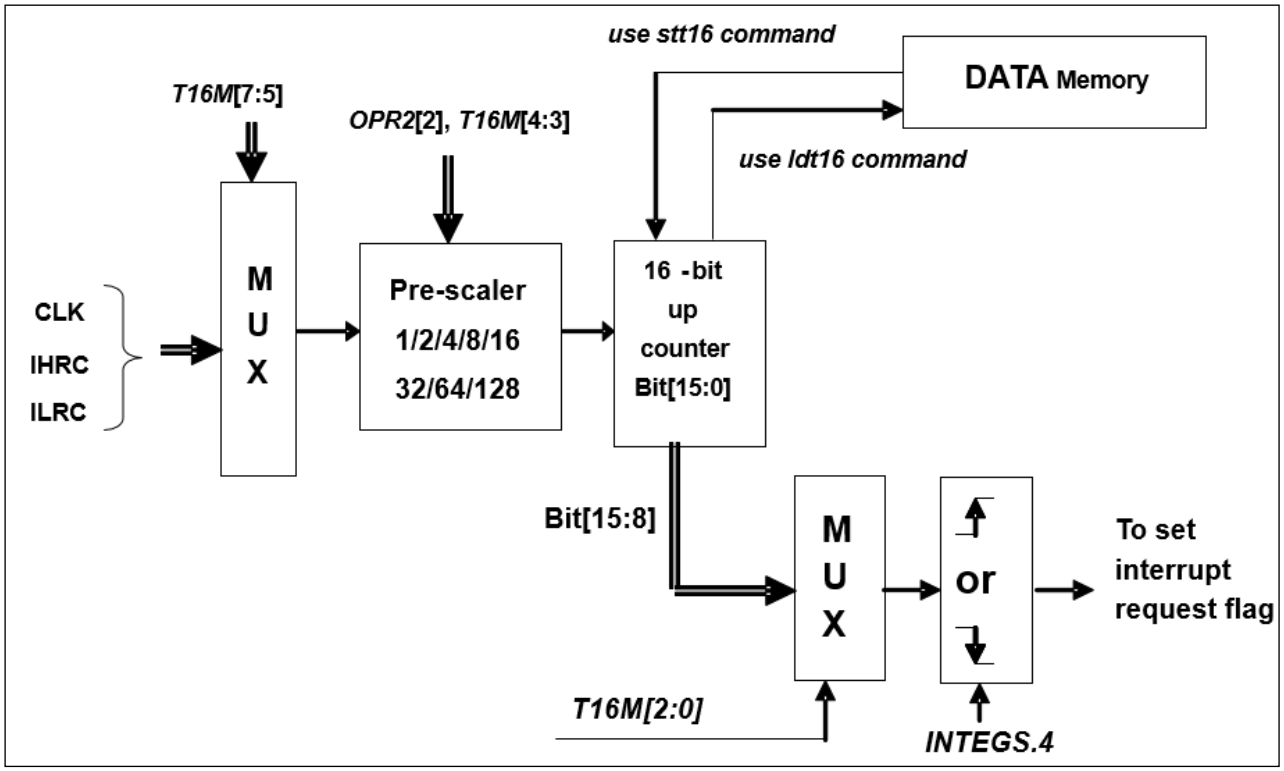


图 15：T16 硬件框图

T16M 共有三个配置参数，第一个参数用来定义 Timer16 的时钟源，第二个参数用来定义预分频器，第三个参数是确定中断源

```
T16M    IO_RW    0x06
$ 7~5:     STOP, SYSCLK, IHRC, ILRC                                  // 1st par.
$ 4~3:     /1, /4, /16, /64(OPR2[2]=0), /2, /8, /32, /128(OPR2[2]=1)  // 2nd par.
$ 2~0:     BIT8, BIT9, BIT10, BIT11, BIT12, BIT13, BIT14, BIT15      // 3rd par.
```

使用者可以依照系统的要求来定义 T16M 参数，例子如下：

```
$    T16M    SYSCLK, /64, BIT15;
// 选择(SYSCLK/64) 当 Timer16 时钟源，每 2^16 个时钟周期产生一次 INTRQ.2=1
// 如果系统时钟 System Clock = IHRC / 4 = 4 MHz
// 则 SYSCLK/64 = 4 MHz/64 = 16 uS，约每 1 S 产生一次 INTRQ.2=1

$    T16M    IHRC, /1, BIT8;
// 选择 IHRC 当 Timer16 时钟源，每 2^9 个时钟周期产生一次 INTRQ.2=1
// 每接收 512 个 IHRC 时钟周期产生一次 INTRQ.2=1

$    T16M    STOP;
// stop Timer16 计数
```

假如 Timer16 是不受干扰自由运行，中断发生的频率可以用下列式子描述：

$$F_{INTRQ_T16M} = F_{clock\ source} \div P \div 2^{n+1}$$

其中，F 是 Timer16 的时钟源频率；

P 是 *OPR2*[2]和 *T16M* [4:3]的选项 (1, 4, 16, 32, 64, 128)

N 是中断要求选择的位，例如：选择位 10，那么 n=10。

10.1.2. Timer16 溢出时间

当设定 *$ INTEGS BIT_R* 时（这是 IC 默认值），且设定 *T16M* 计数器 BIT8 产生中断，若 T16 计数从 0 开始，则第一次中断是在计数到 0x100 时发生（BIT8 从 0 到 1），第二次中断在计数到 0x300 时发生（BIT8 从 0 到 1）。所以设定 BIT8 是计数 512 次才中断。请注意，如果在中断中重新给 T16M 计数器设值，则下一次中断也将在 BIT8 从 0 变 1 时发生。

如果设定 *$ INTEGS BIT_F*（BIT 从 1 到 0 触发）而且设定 T16M 计数器 BIT8 产生中断，则 T16 计数改为每次数到 0x200/0x400/0x600/…时发生中断。两种设定 INTEGS 的方法各有好处，也请注意其中差异。

10.1.3. Timer16 控制寄存器(*T16M*)，地址 = 0x06

位	初始值	读/写	描 述	
7 - 5	000	读/写	Timer16 时钟选择： 000：停用 Timer16 001：CLK 系统时钟 010：保留 011：保留 100：IHRC 101：保留 110：ILRC 111：保留	
4 - 3	00	读/写	*OPR*2.2=0	Timer16 的时钟分频器。00：/1 01：/4 10：/16 11：/64
			*OPR*2.2=1	Timer16 的时钟分频器。00：/2 01：/8 10：/32 11：/128
2 - 0	000	读/写	中断源选择。当选择位由低变高时，发生中断事件。 0：Timer16 位 8 1：Timer16 位 9 2：Timer16 位 10 3：Timer16 位 11 4：Timer16 位 12 5：Timer16 位 13 6：Timer16 位 14 7：Timer16 位 15	
其中，OPR2 是选择寄存器，地址=0x47。				

选项寄存器 2 (*OPR2*)，地址=0x47			
位	初始值	读/写	描 述
7 – 5	-	-	保留位。请设置为 0.
4	0	只写	PWM 同臂保护器。0 / 1：启用 / 停用
3	-	-	保留位。请设置为 0.
2	**0**	只写	**Timer16 时钟预分频器选择选项。请查看 T16M 寄存器。**
1	0	只写	外部中断 1 引脚选择选项。0 / 1：X / PA4
0	0	只写	外部中断 0 引脚选择选项。0 / 1：PA7 / PA4

10.2. 16-bit Timer (Timer4)

10.2.1. Timer4 介绍

MF324 提供另一个 16 位硬件计数器 Timer4，其模块框图如图 16。

定时/计数器 4 的时钟源由寄存器 TM4C[7:5] 选择。在时钟送到 16 位计数器(counter16)之前，可通过寄存器 TM4C[4:2]可对时钟进行预分频处理，有÷1、÷2、÷4、÷8、÷16、÷32、÷64、÷128 等 8 种选项。

当 16 位定时器计数值达到上限寄存器 *TM4BH* 和 TM4BL 设定的范围时，定时器将自动清除为零并产生中断请求。上限寄存器用于定义 Timer4 的周期，使用时需要先写 *TM4BH* 。

16 位计数器的初始值可由寄存器 *TM4CTH* 和 *TM4CTL* 设置。也可以从寄存器中读取 16 位计数器的值。计数器寄存器用于定义定时器 4 的周期，需要先写入 *TM4CTH*。

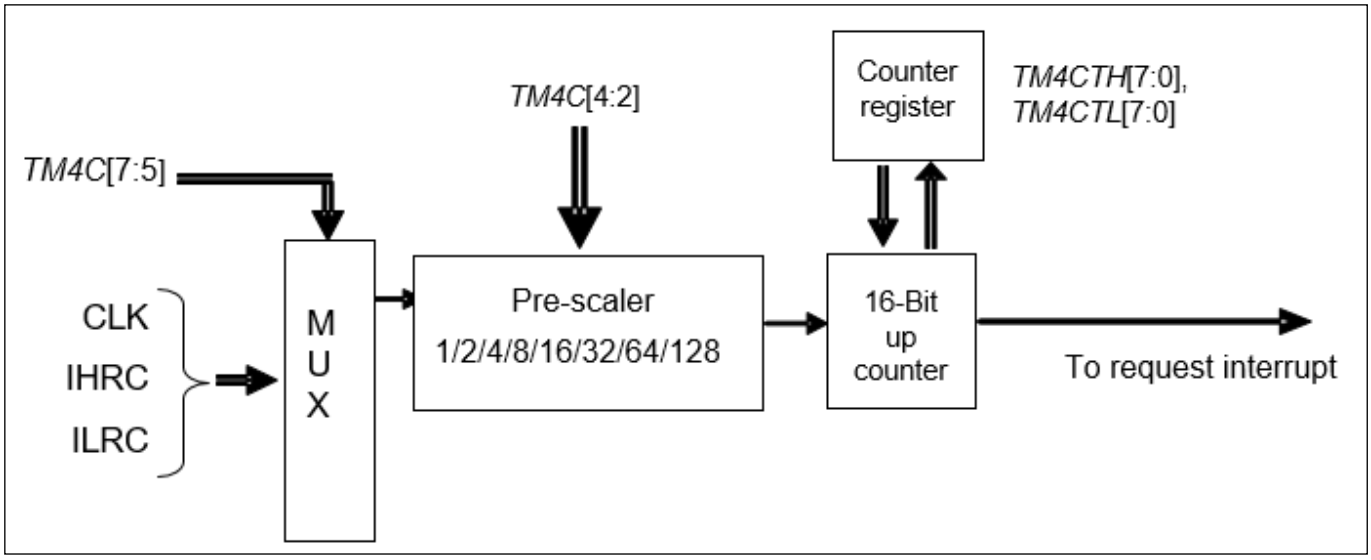


图 16：Timer4 硬件框图

10.2.2. Timer4 模式寄存器 (*TM4C*), 地址=0x22

位	初始值	读/写	描 述
7 - 5	000	只写	Timer 时钟选择： 000：停用 Timer4 001：CLK 系统时钟 010：保留 011：保留 100：IHRC 101：保留 110：ILRC 111：保留
4-2	0000	只写	定时器时钟预分频选择 000: /1 001: /2 010: /4 011: /8 100: /16 101: /32 110: /64 111: /128
1-0	-	只写	保留

10.2.3. Timer4 上限高字节寄存器(TM4CTH)，地址=0x23

位	初始值	读/写	描 述
7 - 0	0x00	读/写	写: 定时/计数器 4 高字节寄存器的初始值 读: 定时/计数器 4 高字节寄存器

10.2.4. Timer4 上限低字节寄存器 (TM4CTL)，地址 = 0x24

位	初始值	读/写	描 述
7 - 0	0x00	读/写	写: 定时/计数器 4 低字节寄存器的初始值 读: 定时/计数器 4 低字节寄存器

10.2.5. Timer4 高字节寄存器 (TM4BH), 地址= 0x25

位	初始值	读/写	描 述
7 - 0	0x00	只写	Timer4 高字节寄存器

10.2.6. Timer4 低字节寄存器 (TM4BL), 地址= 0x26

位	初始值	读/写	描 述
7 - 0	0x00	只写	Timer4 低字节寄存器

©Copyright 2026, PADAUK Technology Co. Ltd

10.3. 8 位计数器 (Timer2, Timer3)

MF324 内置两个 8 位硬件定时器（Timer2/TM2、Timer3/TM3）。因两个 8 位定时器功能一致，以 Timer2 为例说明其功能，图 17 为 Timer2 硬件框图。

TM2C 寄存器的第 7~4 位（Bit [7:4]）用于选择 Timer2 的时钟源。时钟分频模块由 TM2S 寄存器的第 7~5 位（Bit [7:5]）控制，且 TM2S 寄存器的第 4~0 位（Bit [4:0]）为一个分频模块提供 1~32 倍分频系数。通过预分频功能与分频功能的组合，Timer2 的时钟频率（TM2_CLK）可实现宽范围、灵活可调。计数器的值可通过 TM2CT 寄存器进行设置或回读。

Timer2 计数器仅支持 8 位向上计数操作。当 8 位计数器的值达到边界寄存器 TM2B 的设定值时，将自动清零并产生中断请求，该边界寄存器用于定义 Timer2 的周期。

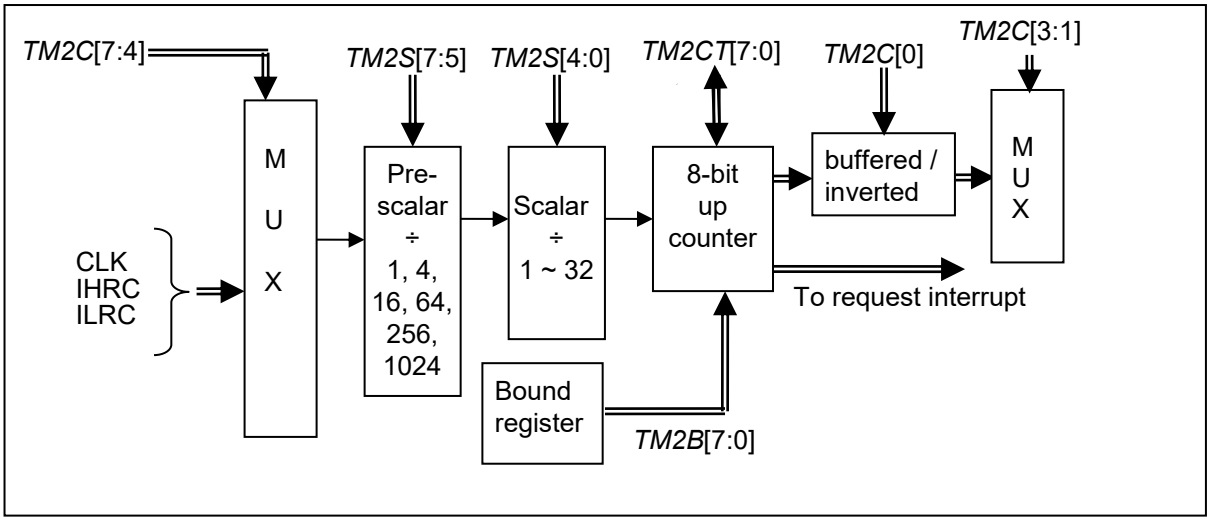


图 17：Timer5 硬件图

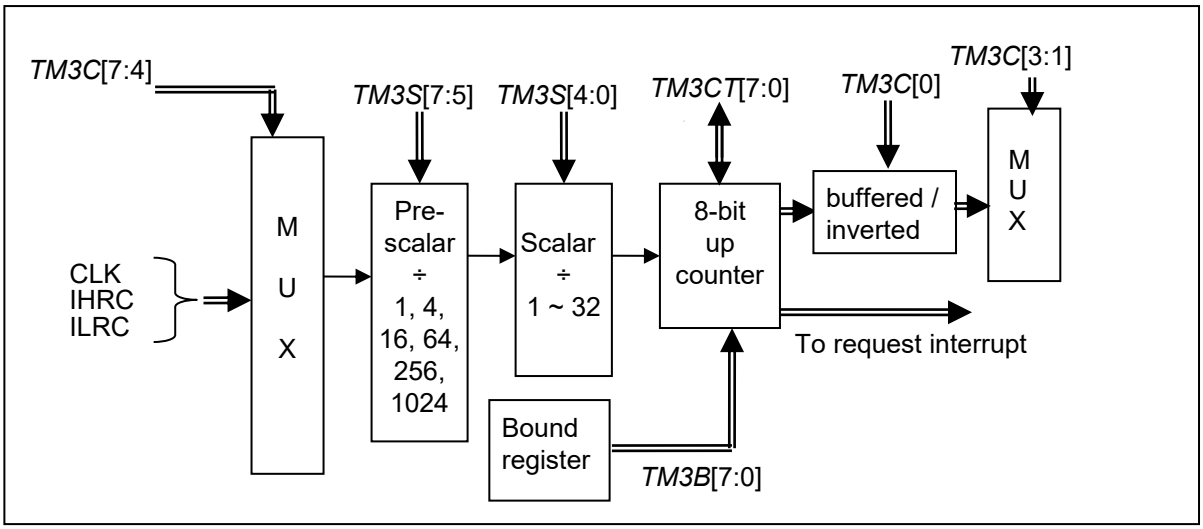


图 18：Timer3 硬件图

10.3.1. Timer2 模式寄存器（*TM2C*），地址= 0x27

位	初始值	读/写	描述
7 - 4	0000	只写	Timer2 时钟源选择。 0000：停用 0001：系统时钟 0010：IHRC 0011：保留 0100：ILRC 0101 - 1111：保留
3 - 0	-	只写	保留。请保持为 0.

10.3.2. Timer2 高字节寄存器（*TM2CT*），地址= 0x28

位	初始值	读/写	描述
7 - 0	0x00	只写	写入：Timer2 计数器寄存器初始值 读取：Timer2 计数器寄存器

10.3.3. Timer2 分频寄存器（*TM2S*），地址= 0x29

位	初始值	读/写	描述
7 - 5	000	只写	Timer2 时钟预分频 000：÷ 1 001：÷ 4 010：÷ 16 011：÷ 64 100：÷ 256 101：÷ 1024 11X：保留
4 - 0	00000	只写	Timer2 时钟分频

10.3.4. Timer2 上限寄存器 （*TM2B*），地址 = 0x2A

位	初始值	读/写	描述
7 - 0	0x00	只写	Timer2 上限寄存器

10.3.5. Timer3 控制寄存器 （*TM3C*），地址 = 0x2B

位	初始值	读/写	描述
7 - 4	0000	读/写	Timer3 时钟选择 0000：停用 0001：系统时钟 0010：IHRC 0011：保留 0100：ILRC 0101 - 1111 ：保留
3 - 0	-	-	保留。请保持为 0

10.3.6. Timer3 上限寄存器（*TM3CT*），地址= 0x2C

位	初始值	读/写	描述
7 - 0	0x00	读/写	写操作：Timer3 计数器寄存器初始值 读操作：Timer3 计数器寄存器

10.3.7. Timer3 分频寄存器（*TM3S*），地址= 0x2D

位	初始值	读/写	描述
7 - 5	000	只写	Timer3 时钟与分频 000：÷ 1 001：÷ 4 010：÷ 16 011：÷ 64 100：÷ 256 101：÷ 1024 11X：保留
4 - 0	00000	只写	Timer3 时钟分频

10.3.8. Timer3 上限寄存器(*TM3B*)，地址= 0x2E

位	初始值	读/写	描述
7 - 0	0x00	只写	Timer3 上限寄存器

10.4. 12 位 PWM 产生器

MF324 内部集成了三个 12 位硬件 PWM 发生器，具有可编程周期和占空比，并支持死区控制。PWM 配置为互补（中心）模式。时钟源可通过 PWMGC[3:2]选择时钟源、系统时钟、IHRC 或 IHRC*2。内置预分频逻辑，可通过寄存器 PWMGCS[1:0]选择 1、2、4、8 倍分频。集成基本同相保护硬件。

此外，它还具备灵活的控制组合，以支持三相无刷直流电机。

注意：如果 PWMGC 与源自 IHRC 的时钟源配合使用，软件必须在禁用 IHRC（**CLKMD.En_IHRC = 0**）或执行 **STOPSYS** 指令之前，先禁用 PWMGC。

10.4.1. 带死区 PWM 硬件和时序框图

MF324 内置带死区控制的 PWM 发生器，图 19 为其硬件框图。PWM 波形的周期由 PWM 上边界寄存器 PWMUpBH（高字节）与 PWMUpBL（低字节）定义；占空比由 PWM0DtH（高字节）与 PWM0DtL（低字节）寄存器定义；死区时间由 PWMDdZVF 与 PWMDdZVR 寄存器定义。PWM 输出信号可通过 PWMGC1 与 PWMGC2 寄存器切换为高电平 / 低电平 / PWM + / PWM - 。当 IO 引脚 PBX 输出 PWM 信号时，可通过 PWMPBC1 与 PWMPBC2 寄存器进行切换控制。PWM 可通过 PWMADCH 与 PWMADCL 寄存器配置触发 ADC 转换。

注意：需要先设置 PWMGC1，在设置 PWMGC2 后寄存器才会生效。

MF324 的 PWM 模块内置紧急停止专用电路，具体配置请参考 MISC 寄存器的第 7~6 位（MISC [7:6]）。

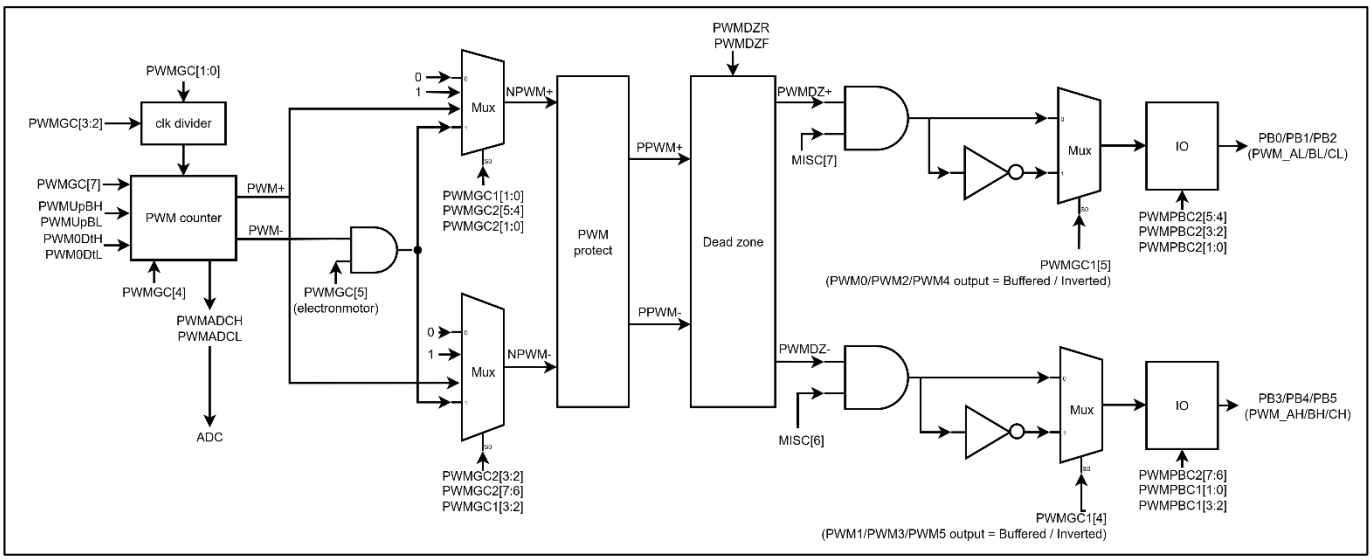


图 19：12 位 PWM 生成器硬件框图

10.4.2. 时序图

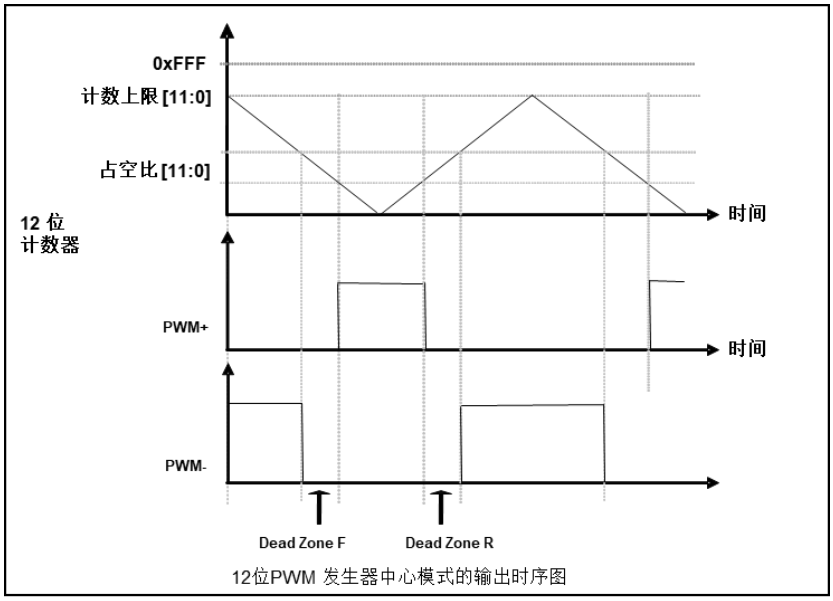


图 20：12 位 PWM 发生器中心模式的输出时序图

10.4.3. 12 位 PWM 计算公式

如果 F_{IHRC} 代表 IHRC 振荡器的频率且 IHRC 被选为 12 位 PWM 的时钟源，则它的频率和占空比可由以下公式得出：

PWM 输出频率 $F_{PWM} = F_{IHRC} \div [P \times CB]$

PWM 占空比（时间）$= (1/F_{IHRC}) * [DB \div CB]$

这里， *PWMGC*[1:0] = P；预分频

12-bit Duty_Bound[11:0] = {pwmdth[7:0],pwmdtl[7:4]} = **DB**;占空比

11-bit Counter_Bound[11:1] X2 = {pwmcubh[7:0], pwmcubl[7:5]} X2 = **CB**; 计数器

10.4.4. 12 位 PWM 相关寄存器

10.4.4.1. PWM PB 控制寄存器 1(*PWMPBC1*)，地址= 0x18

位	初始值	读/写	描 述
7-4	-	-	保留
3-2	10	读/写	PB5 输出 PWM1/PWM3/PWM5 类型。 00：PWM1 01：PWM3 10：PWM5 11：保留
1-0	01	读/写	PB4 输出 PWM1/PWM3/PWM5 类型。 00：PWM1 01：PWM3 10：PWM5 11：保留

10.4.4.2. PWM PB 控制寄存器 2(*PWMPBC2*)，地址= 0x19

位	初始值	读/写	描 述
7 - 6	00	读/写	PB3 输出 PWM1/ PWM3/ PWM5 类型 00：PWM1 01：PWM3 10：PWM5 11：保留
5 - 4	10	读/写	PB2 输出 PWM0/ PWM2/ PWM4 类型 00：PWM0 01：PWM2 10：PWM4 11：保留
3 - 2	01	读/写	PB1 输出 PWM0/ PWM2/ PWM4 类型。 00：PWM0 01：PWM2 10：PWM4 11：保留
1 - 0	00	读/写	PB0 输出 PWM0/ PWM2/ PWM4 类型。 00：PWM0 01：PWM2 10：PWM4 11：保留

©Copyright 2026, PADAUK Technology Co. Ltd

10.4.4.3. PWM 发生器控制寄存器 1（*PWMGC1*）地址 = 0x1A

位	初始值	读/写	描 述
7-6	-	-	保留
5	0	读写	PWM0/PWM2/PWM4 输出正相/反相。 0：正相 1：反相
4	0	读写	PWM1/PWM3/PWM5 输出正相/反相。 0：正相 1：反相
3 - 2	00	读写	PWM5 输出类型。 00：强制低电平 01：PWM+ 10：PWM- 11：强制高电平
1-0	00	读写	PWM4 输出类型。 00：强制低电平 01：PWM+ 10：PWM- 11：强制高电平

10.4.4.4. PWM 发生器控制寄存器 2 (PWMGC2)，地址= 0x1B

位	初始值	读/写	描 述
7 - 6	00	读写	PWM3 输出类型。 00：强制低电平 01：PWM+ 10：PWM- 11：强制高电平
5 - 4	00	读写	PWM2 输出类型 00：强制低电平 01：PWM+ 10：PWM- 11：强制高电平
3 - 2	00	读写	PWM1 输出类型 00：强制低电平 01：PWM+ 10：PWM- 11：强制高电平
1 - 0	00	读写	PWM0 输出类型 00：强制低电平 01：PWM+ 10：PWM- 11：强制高电平

©Copyright 2026, PADAUK Technology Co. Ltd

10.4.4.5. PWM 发生器控制寄存器 (PWMGC)，地址 = 0x1C

位	初始值	读/写	描 述
7	0	读写	启用 PWM 发生器。 0 / 1: 停用 / 启用
6	0	只读	PWMG0(PWM+)的输出
5	0	读写	PWM-禁用电机模式 0: 停用 1: 启用
4	0	只写	PWM 计数器复位 向该位写入 “1” 可清除 PWM 计数器，且计数器复位后，此位将自动清零为 0。
3 - 2	00	读写	PWM 发生器时钟源 00 : 系统时钟 01 : IHRC 10 : IHRC*2 11 : 保留
1 - 0	00	读写	PWMGx 时钟预分频 00 : ÷ 1 01 : ÷ 2 10 : ÷ 4 11 : ÷ 8

10.4.4.6. 杂项设置寄存器 (MISC)，地址= 0x34

位	初始值	读/写	描 述
7	0	只写	启用 PWM0/PWM2/PWM4 输出紧急停止功能。0 / 1：停用 / 启用
6	0	只写	启用 PWM1/PWM3/PWM5 输出紧急停止功能。. 0 / 1：停用 / 启用
5 - 3	-	-	保留。请保持 0。
2			保留。请保持 0。
1 - 0	10	只写	看门狗超时周期。 01: 4K ILRC 10: 16K ILRC

10.4.4.7. PWM 计数器上限高位寄存器 (PWMUPBH)，地址= 0x3D

位	初始值	读/写	描 述
7 - 0	8’h00	只写	PWM 计数器上限的位[11:4]

10.4.4.8. PWM 计数器上限低位寄存器(PWMUPBL)，地址 = 0x3E

位	初始值	读/写	描 述
7 - 5	3’h0	只写	PWM0 生成器占空比值位[3:1]
4 - 0	-	-	保留

10.4.4.9. PWMG0 占空比高位寄存器 (PWM0DTH)，地址 = 0x3F

位	初始值	读/写	描 述
7 - 0	8'h00	只写	PWM0 生成器占空比值位[11:4]

10.4.4.10. PWMG0 占空比低位寄存器(PWM0DTL)，地址 = 0x40

位	初始值	读/写	描 述
7 - 4	4'h0	只写	PWM0 生成器占空比值位[3:0]
3 - 0	-	-	保留

10.4.4.11. PWMG ADC 触发高位寄存器 (PWMADCH)，地址 = 0x41

位	初始值	读/写	描 述
7 - 0	8'h00	只写	PWM2 生成器占空比值位[11:4]

10.4.4.12. PWMG ADC 触发低位寄存器 (PWMADCL)，地址 = 0x42

位	初始值	读/写	描 述
7 - 4	4'h0	只写	PWM2 生成器占空比值位[11:4]
3 - 0	-	-	保留

10.4.4.13. PWM 前置死区寄存器(*PWMDDZVF*)，地址 = 0x43

位	初始值	读/写	描 述
7 - 1	-	只写	PWM 发生器的前置死区值 位[7:1]。
0	-	-	保留

10.4.4.14. PWM 后置死区寄存器 (*PWMDDZVR*)，地址= 0x44

位	初始值	读/写	描 述
7 - 1	-	只写	PWM 发生器后置死区值的位[7:1]。
0	-	-	保留

11. 省电与掉电

MF324 有三个由硬件定义的操作模式，分别为：正常工作模式，电源省电模式和掉电模式。正常工作模式是所有功能都正常运行的状态，省电模式(stopexe)是在降低工作电流而且 CPU 保持在随时可以继续工作的状态，掉电模式(stopsys)是用来深度的节省电力。因此，省电模式适合在偶尔需要唤醒的系统工作，掉电模式是在非常低消耗功率且很少需要唤醒的系统中使用。

11.1. 省电模式 (“stopexe”)

使用 ***stopexe*** 指令进入省电模式，只有系统时钟被停用，其余所有的振荡器模块都仍继续工作. 所以只有 CPU 是停止执行指令，然而，对 Timer16 计数器而言，如果它的时钟源不是系统时钟，那 Timer16 仍然会保持计数。省电模式下，唤醒源可以是 IO 的切换，或者 Timer16 计数到设定值时（假如 Timer16 的时钟源是 IHRC 或者 ILRC），或比较器唤醒（需同时设定 GPCC.7 为 1 与 GPCS.6 为 1 来启用比较器唤醒功能）。假如系统唤醒是因输入引脚切换，那可以视为系统继续正常运行。 省电模式的详细信息如下所示：

- IHRC 和 EOSC 振荡器模块：没改变，如果被启用，则仍然保持运行状态；
- ILRC 振荡器模块：必须保持启用，唤醒时需要靠 ILRC 启动。
- 系统时钟：停用，因此 CPU 停止运行；
- MTP 存储器关闭；
- Timer 计数器：：若 Timer 计数器的时钟源是系统时钟或其相应的时钟振荡器模块被停用，则 Timer 停止计数；否则，仍然保持计数。（其中，Timer 包含 Timer16，TM2，TM3）
- 唤醒来源：
  a. IO Toggle 唤醒： IO 在数字输入模式下的电平变换（*PxC* 位是 0，*PxDIER* 位是 1）
  b. Timer 唤醒：如果计数器 (Timer)的时钟源不是系统时钟，则当计数到设定值时，系统会被唤醒，它在上升沿和下降沿都会被唤醒。
  c. 比较器唤醒：使用比较器唤醒时，需同时设定 *GPCC*.7 为 1 与 *GPCS*.6 为 1 来启用比较器唤醒功能。但请注意：内部 1.20V Bandgap 参考电压不适用于比较器唤醒功能。

以下例子是利用 Timer16 来唤醒系统因 ***stopexe*** 的省电模式：

```
$ T16M    ILRC, /1, BIT8          // Timer16 设置
…
WORD      count     =     0;
STT16     count;
stopexe;
…
```

Timer16 的初始值为 0，在 Timer16 计数了 256 个 ILRC 时钟后，系统将被唤醒。

11.2. 掉电模式（“stopsys”）

掉电模式是深度省电的状态，所有的振荡器模块都会被关闭。通过使用“stopsys” 指令，芯片会直接进入掉电模式。在下达 stopsys 指令之前建议将 GPCC.7 设为 0 来关闭比较器。下面显示发出 stopsys 命令后，如下是 PFS122B 内部详细的状态：

- 所有的振荡器模块被关闭；
- MTP 存储器被关闭；
- SRAM 和寄存器内容保持不变；
- 唤醒源：IO 在数字输入模式下电平变换（PxDIER 位是 1）

通过输入引脚触发的唤醒可视为正常程序执行的延续。为了最大程度降低功耗，在进入掉电模式之前，应仔细配置所有 I/O 引脚的状态。如果使用了 PWM 功能，建议在执行 “stopsys” 指令前禁用 PWM，并在唤醒后重新使能。掉电模式的参考示例程序如下所示：

```
$PWMGC Disable ;
CLKMD   =   0xF4;        //   Change clock from IHRC to ILRC
CLKMD.4 =   0;           //   disable IHRC
…
while (1)
  {
        STOPSYS;         //   enter power-down
        if  (…)  break;  //   if wakeup happen and check OK, then return to high speed,
                         //   else stay in power-down mode again.
  }
CLKMD   =   0x34;        //   Change clock from ILRC to IHRC/2
$PWMGC Enable;
```

11.3 唤醒

进入掉电或省电模式后，PFS122B 可以通过切换 IO 引脚恢复正常工作，而 Timer 的唤醒只适用于省电模式。表 5 显示 *stopsys* 掉电模式和 *stopexe* 省电模式在唤醒源的差异。

掉电模式（stopsys）和省电模式（stopexe）在唤醒源的差异			
	IO 引脚切换	计时器中断	比较器唤醒
STOPSYS	是	否	否
STOPEXE	是	是	是

表 5：掉电模式和省电模式在唤醒源的差异

当使用 IO 引脚来唤醒 PFS122B， *pxdier* 寄存器应对每一个相应的引脚正确设置“使能唤醒功能”。从唤醒事件发生后开始计数，正常的唤醒时间大约是 3000 个 ILRC 时钟周期，另外，PFS122B 提供快速唤醒功能，透过 *misc* 寄存器选择快速唤醒大约 45 个 ILRC 时钟周期。

模式	唤醒模式	切换 IO 引脚的唤醒时间(t_{WUP})
STOPEXE 省电模式 STOPSYS 掉电模式	正常唤醒	3000* T_{ILRC}, 这里的 T_{ILRC} 是指 ILRC 时钟周期

12. 特殊功能

12.1. 通用比较器

12.1.1. 通用比较器硬件框图

MF324 内置一个硬件比较器，该比较器可用于比较两个引脚间的信号，也可将信号与内部参考电压 $V_{internal\ R}$ 或外部 PA5 引脚电压进行比较。待比较的两路信号中，一路作为比较器的正输入端，另一路作为比较器的负输入端。对于通用比较器，其正输入引脚由寄存器 GPCC 的第 0 位（GPCC.0）选择，负输入引脚则由寄存器 GPCC 的第 1~3 位（GPCC [3:1]）选择。

比较器输出的结果可以：

(1) 可通过 GPCC 寄存器的第 6 位（GPCC.6）回读（该状态）。

(2) 可通过 GPCC 寄存器的第 4 位（GPCC.4）反转（输出）极性。

(3) 可通过 INTEGS 寄存器的第 5~6 位（INTEGS [6:5]）选择中断触发沿。

上电复位后，该比较器处于禁用状态，将 GPCC 寄存器的第 7 位（GPCC.7）置 1 即可启用该比较器。

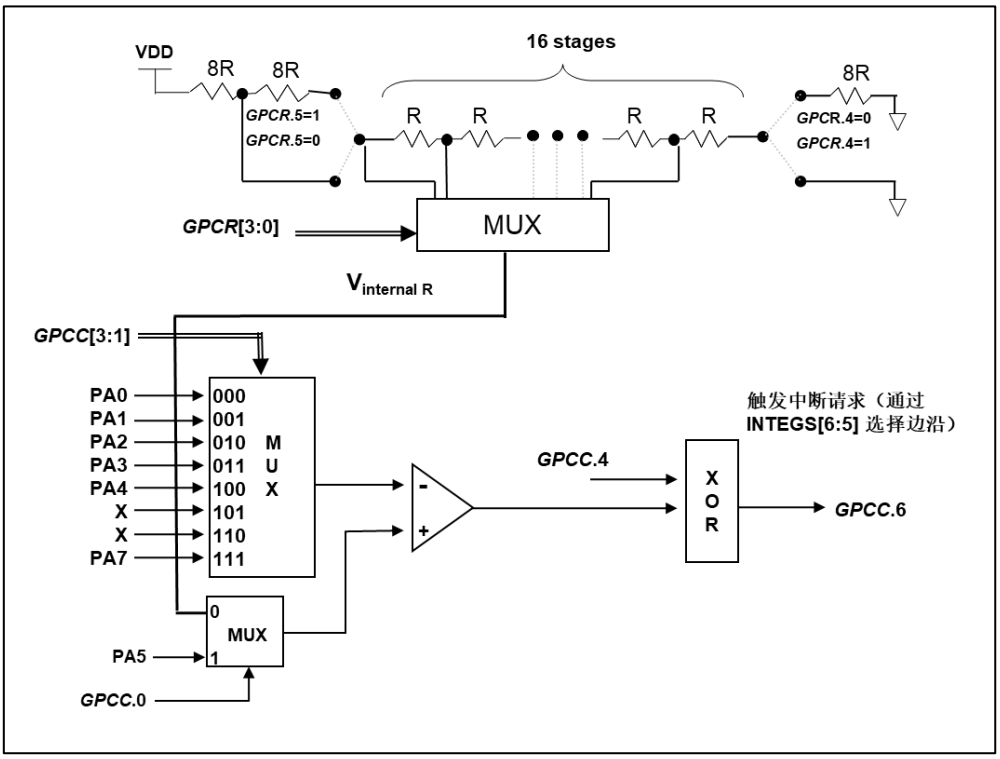


图 21：比较器 1 硬件框图

12.1.2. 通用比较器控制寄存器(GPCC)，地址= 0x15

位	初始值	读/写	描 述
7	0	读/写	启用比较器 0 / 1 ：停用/启用 当此位被设置为启用，请同时设置相应的模拟输入引脚是数字停用，以防止漏电。
6	-	只读	比较器结果 0：正输入 < 负输入 1：正输入 > 负输入
5	-	-	保留
4	0	读/写	反转比较器 输出结果的极性 0：极性不反转 1：极性反转
3 - 1	000	读/写	选择通用比较器负输入的来源 000：PA0 001：PA1 010：PA2 011：PA3 100：PA4 101：保留 110：保留 111：PA7
0	0	读/写	选择比较器正输入的来源 0：$V_{internal\ R}$ 1：PA5

12.1.3. 通用比较器结果寄存器(GPCR)，地址= 0x16

位	初始值	读/写	描 述
7	0	只写	GPC 比较结果输出至 PA6 引脚 0：停用 1：启用
6	-	-	保留
5	0	只写	选择通用比较器高量程
4	0	只写	选择通用比较器低量程
3 - 0	0000	只写	选择通用比较器的电压级别 0000（最低）~ 1111（最高）

12.1.4. 模拟讯号输入

图 22 展示了一个简化的模拟输入电路。通用比较器的所有模拟输入引脚均与数字输入引脚共享功能；该引脚配有连接至 VDD 和 GND 的反向偏置 ESD（静电释放）保护二极管。因此，模拟输入信号必须保持在 VDD 和 GND 之间。

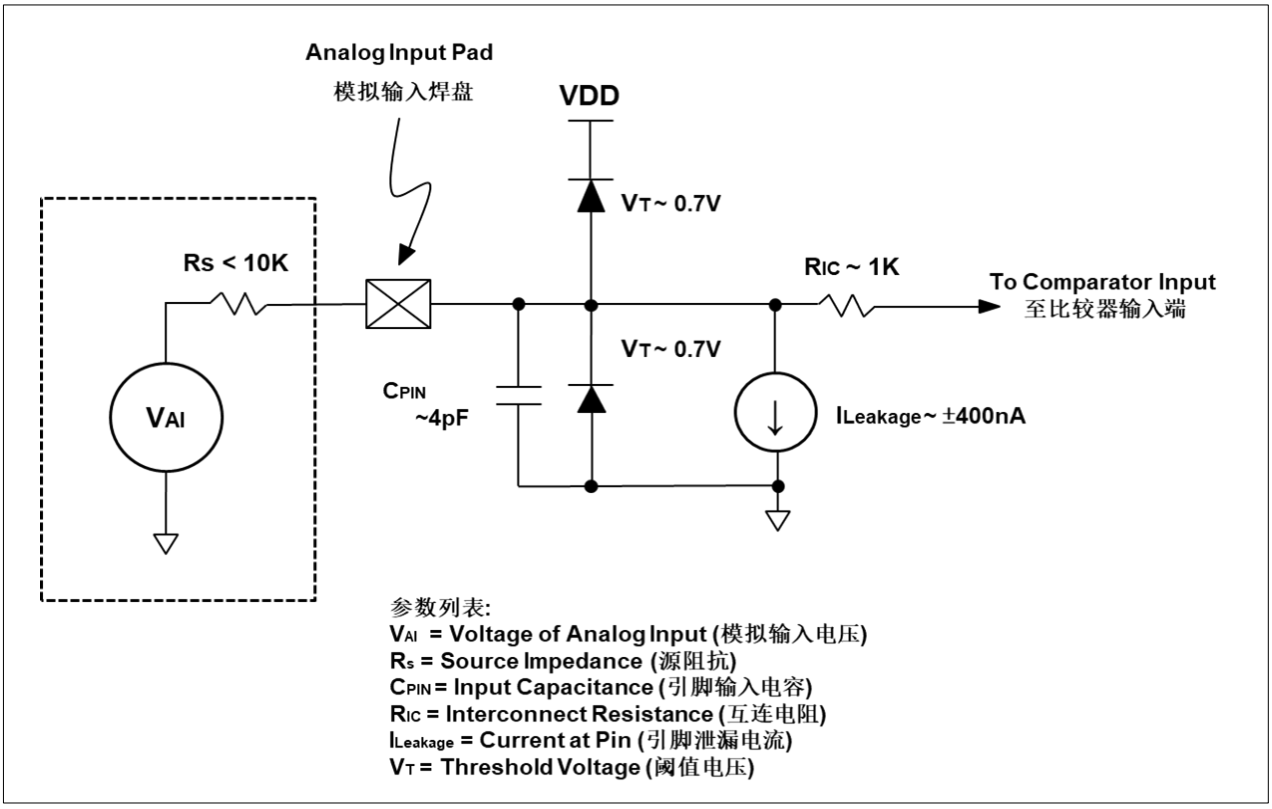


图 22：通用比较器模拟输入引脚模型

12.1.5. 内部参考电压 ($V_{internal\ R}$)

内部参考电压 $V_{internal\ R}$ 是通过串联电阻构建的，以提供不同电平的参考电压。GPCR 寄存器的第 4 位和第 5 位用于选择 $V_{internal\ R}$ 的最大值和最小值，而 GPCR[3:0]则用于从定义的最大电平到最小电平之间按 16 等分的电压电平中选择其中一个。通过设置 GPCR 寄存器，内部参考电压 $V_{internal\ R}$ 的范围可以从(1/32)*VDD 到(3/4)*VDD。

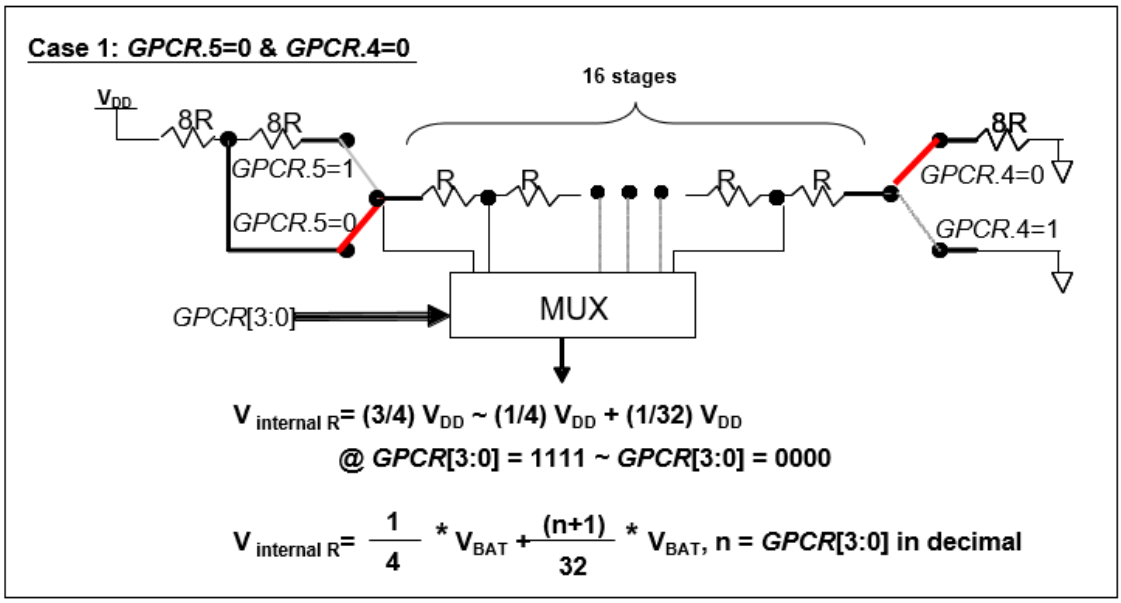


图 23：$V_{internal\ R}$ 硬件接法 (*GPCR*.5=0 and *GPCR*.4=0)

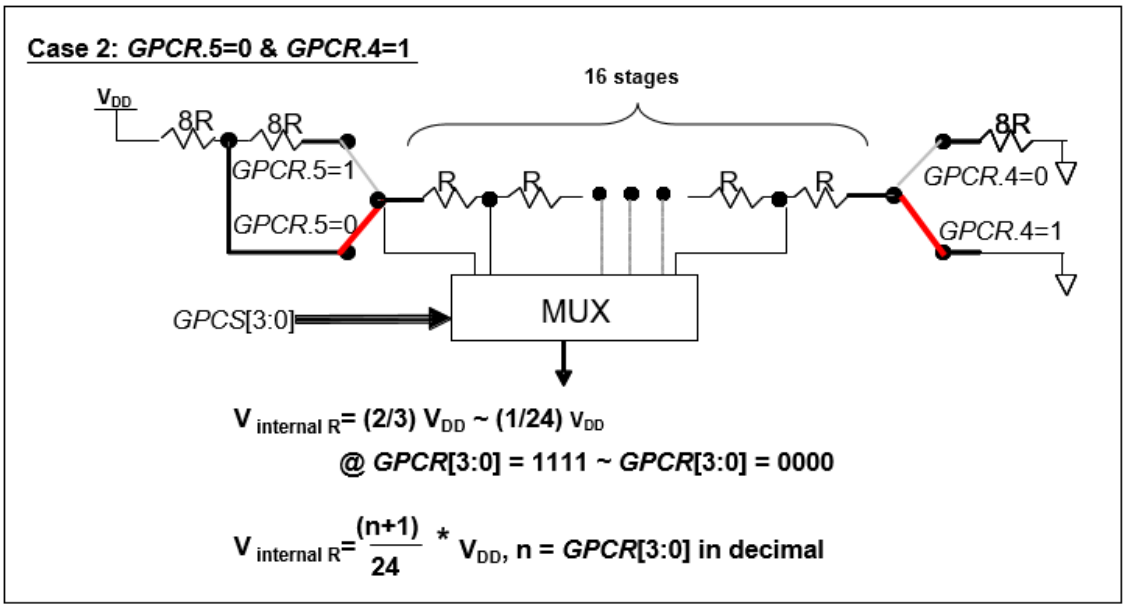


图 24：$V_{internal\ R}$ 硬件接法 (*GPCR*.5=0 and *GPCR*.4=1)

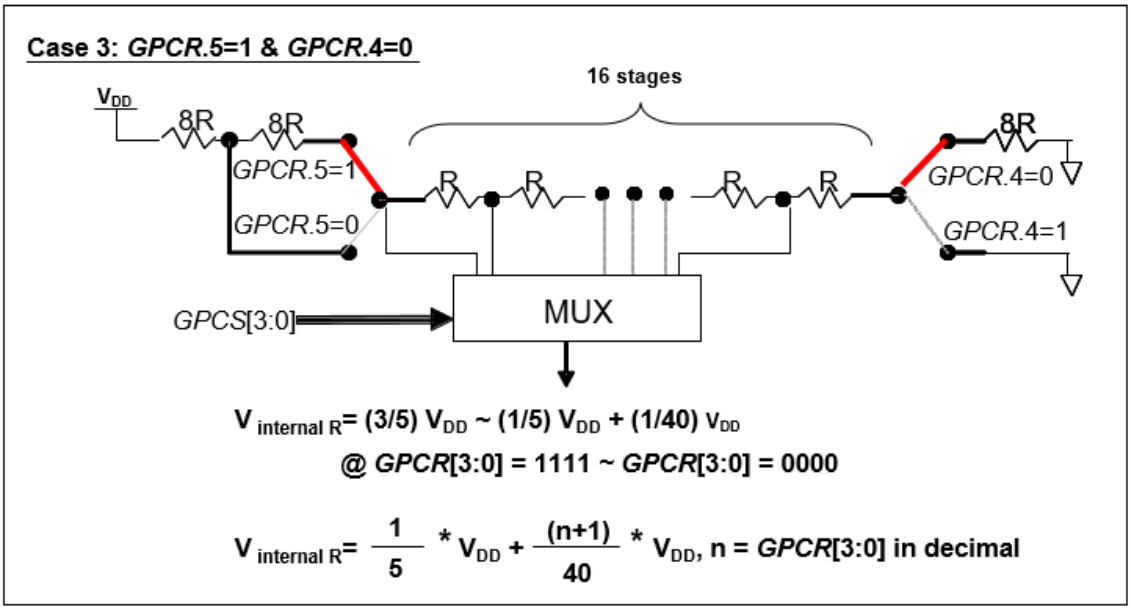


图 25：$V_{internal\ R}$ 硬件接法(*GPCR*.5=1 and *GPCR*.4=0)

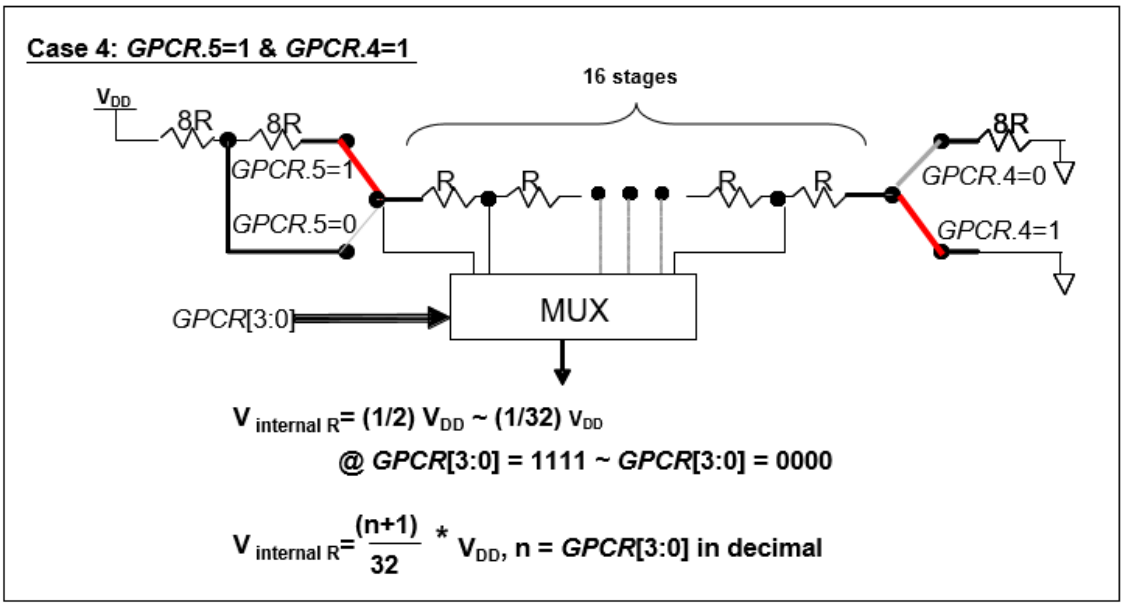


图 26：$V_{internal\ R}$ 硬件接法（*GPCR*.5=1 and *GPCR*.4=1）

12.1.6. 使用比较器

例 1：

选择 PA0 为负输入和 $V_{internal\ R}$ 的电压为(18/32)*VDD 作为正输入。$V_{internal\ R}$ 选择上图 *GPCR*[5:4] = 2b'00 的配置方式，*GPCR*[3:0] = 4b'1001 (n=9) 以得到 $V_{internal\ R}$ = (1/4)*VDD + [(9+1)/32]*VDD _ = [(9+9)/32]*VDD = (18/32)*VDD 的参考电压。

```
GPCC    = 0b1_0_0_0_000_0;     // 启用 comp，输入 -：PA0，输入 +：Vinternal R、
GPCR    = 0b00_0_0_1001;       //  Vinternal R = VDD*(18/32)
PADIER  = 0bxxxx_xxx_0;        // 关闭 PA0 数字输入，以防止漏电流
```

例 2：

如果将 $V_{internal\ R}$ 设置为正输入（22/40）*VDD 电压电平，并将 PA2 设置为负输入，则比较器的结果将反转，且不会输出到 PA2。将 $V_{internal\ R}$ 配置为上图中的 "GPCR[5:4] = 2b'10 "和 GPCR[3:0] = 4b'1101 (n=13)，则 $V_{internal\ R}$ = (1/5)*VDD + [(13+1)/40]*VDD = [(13+9)/40]*VDD = (22/40)*VDD.

```
GPCC    = 0b1_0_0_1_101_0;     // 启用 comp，反向输出，输入 - : PA2，输入 + : Vinternal R、
GPCR    = 0b00_1_0_1101;       //  Vinternal R = VDD*(22/32)
PADIER  = 0bxxx_x_0_xx;        // 停用 PA2 数字输入，以防止漏电流
```

12.2. 模拟-数字转换器 (ADC) 模块

MF324 提供了一个 12 位分辨率的模数转换模块，该模块分别有 8 个外部通道和 3 个内部信道，内部通道分别用于采集带隙基准电压、反电动势（BEMF）以及四分之一电源电压。

在转换过程中，模拟信号首先会被采样并保持，然后送入转换器，通过逐次逼近法生成转换结果。该 ADC 的模拟参考高电平为电源正电压（VDD），参考低电平始终为地（GND）。

建议在电源电压（VDD）和地（GND）之间放置大于 1 uF 的电容，以获得更好的模数转换结果。

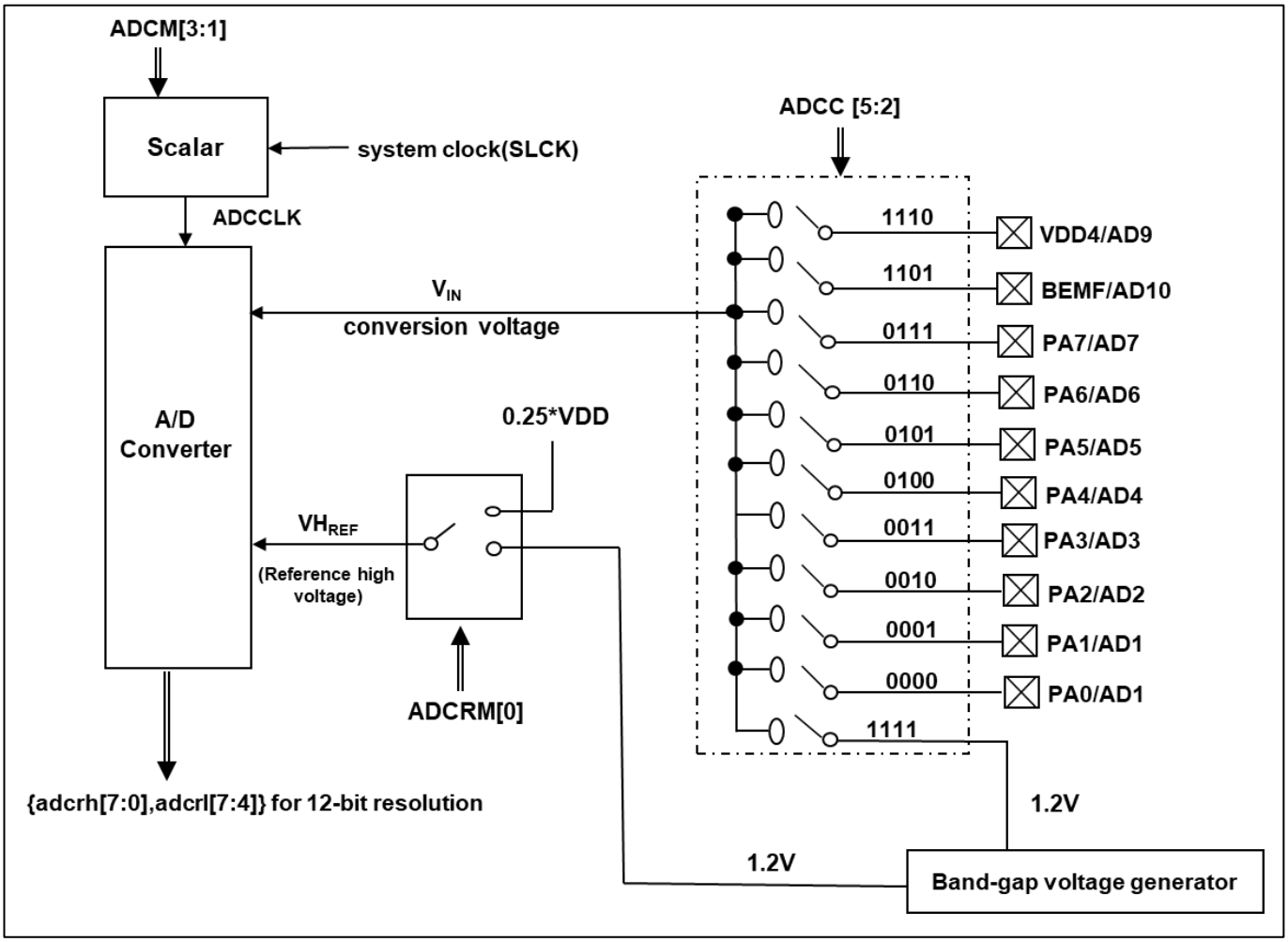


图 27：ADC 模块框图

12.2.1. AD 转换的输入要求

对于模数转换而言，为满足其规定的精度要求，电荷保持电容（C_{HOLD}）必须能够完全充电至参考高电压电平（VDD），并放电至参考低电压电平（GND）。模拟输入模型如图 28 所示，信号驱动源阻抗（RS）和内部采样开关阻抗（R_{ON}）会直接影响给电容 C_{HOLD} 充电所需的时间。内部采样开关阻抗可能会随模数转换器（ADC）的电源电压而变化；信号驱动源阻抗则会影响模拟输入信号的精度。用户必须确保在采样前被测信号是稳定的；因此，最大信号驱动源阻抗在很大程度上取决于待测信号的频率。在输入频率为 500kHz 的情况下，模拟驱动源的推荐最大阻抗约为 10KΩ。

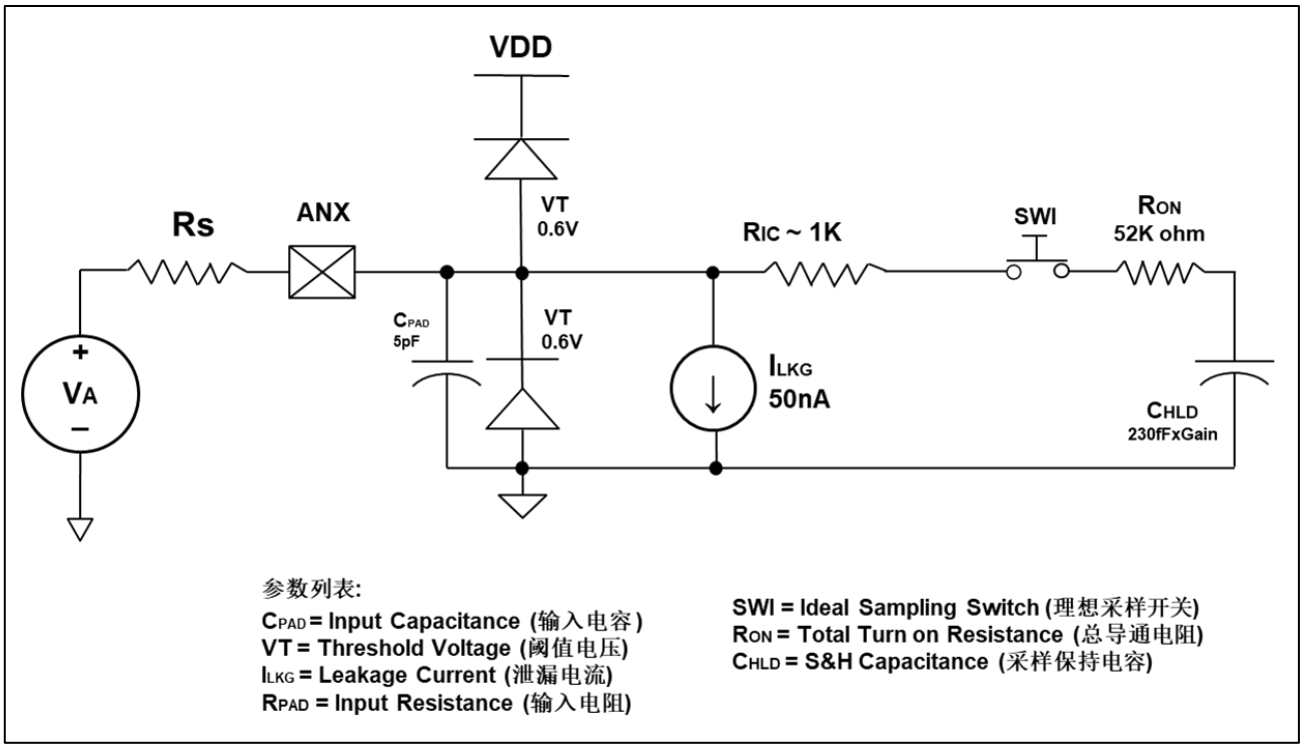


图 28：模拟输入模型

在使用 AD 转换之前，必须确认所选的模拟输入信号的采集时间应符合要求，MF324 系列 ADC 的信号采集时间(T_{ACQ})固定为 ADCLK 的一个时钟周期，ADCLK 的选择必须满足最短信号采集时间。

12.2.2. ADC 时钟选择

ADC 模块的时钟(ADCLK)能够通过 *ADCM* 寄存器来选择，ADCLK 从 CLK÷1 到 CLK÷128 一共有 8 个选项可被选择（CLK 是系统时钟）。由于信号采集时间 T_{ACQ} 是 ADCLK 的一个时钟周期，所以 ADCLK 必须满足这要求，建议 ADC 时钟周期是 0.2us。

12.2.3. AD 转换

AD 转换的过程，从设置 START/DONE(ADCC.6) 为高开始，START/DONE 的标志位内部将会自动清零，然后转换模拟信号将会一位一位的转换，当 AD 转换完成时，START/DONE 将自动置高表示完成转换。当 ADCLK 被选定后，ADCLK 的周期是 T_{ADCLK}，而 AD 转换的时间将是如下：

◆ 12 位分辨率：AD 转换时间 = 16 T_{ADCLK}

12.2.4. 配置模拟引脚

有 11 模拟信号可以被 AD 转换选择：8 路来自外部引脚的模拟输入信号，一路来自 Bandgap 参考电压，1 路来自反电动势（BEMF），另 1 路来自 1/4VDD 电压，为了避免漏电，这些引脚在使用时定义为模拟输入并应停用数字输入功能（设置 *PADIER* 寄存器的相应位为 0）。

ADC 的测量信号属于小信号，为避免测量信号在测量期间被干扰，被选定的引脚应：

(1) 设为输入模式
(2) 关闭弱上拉电阻
(3) 通过端口 A/B 寄存器（*PADIER* / *PBDIER*）设置模拟输入并关闭数字输入

建议使用以下步骤来执行 AD 转换过程：

(1) 配置 ADC 模块：

- ◆ 通过 *ADCM* 寄存器将电压基准配置为高电平，建议采用 Bandgap 模式运行。
- ◆ 通过 *ADCC* 寄存器选择 ADC 输入通道
- ◆ 通过 *ADCM* 寄存器配置 ADC 转换时钟
- ◆ 通过 *PADIER* 寄存器配置所选定的引脚作为模拟输入
- ◆ 通过 *ADCC* 寄存器启用 ADC 模块

(2) 配置 ADC 模块的中断：（如果需要）

- ◆ 清零 *INTRQ* 寄存器第 3 位的 ADC 中断请求标志
- ◆ 启用 *INTEN* 寄存器第 3 位的 ADC 中断请求
- ◆ 通过 *ENGINT* 指令启用全局中断

(3) 启动 ADC 转换：

- ◆ 通过 *ADCC* 寄存器置位 ADC 转换过程控制位启动转换（set1 *ADCC.6*）

(4) 等待完成 AD 转换标志位置位，方法可以用如下的任一种：

- ◆ 通过使用指令“*wait1 ADCC.6*”来等待完成标志；或等待 ADC 的中断

(5) 读取 ADC 的数据寄存器：

- ◆ 读取 *ADCRH* 和 *ADCRL* 数据寄存器

(6) 下一个转换，依要求转到步骤 1 或步骤 2

12.2.5. 使用 ADC

下面的示例演示使用 PA3～PA4 来当 ADC 输入引脚。

首先，定义所选择的引脚：

```
PAC     =   0B_XXX0_0XXX;       //  PA3 ~ PA4 作为输入
PAPH    =   0B_XXX0_0XXX;       //  PA3 ~ PA4 没有弱上拉电阻
PADIER  =   0B_XXX0_0XXX;       //  PA3 ~ PA4 关闭数字输入
```

下一步，设定 *ADCC* 寄存器，示例如下：

```
$   ADCC    Enable, PA3;        //  设置 PA3 作为 ADC 输入
$   ADCC    Enable, PA4;        //  设置 PA4 作为 ADC 输入
```

下一步，设定 *ADCM* 和 *ADCRGC* 寄存器，示例如下：

```
$   ADCM    /16;                //  建议 /16 @系统时钟=8MHz
$   ADCM    /8;                 //  建议 /8 @系统时钟=4MHz
```

接着，开始 ADC 转换：

```
AD_START  =   1;                //  开始 ADC 转换
while (! AD_DONE)   NULL;       //  等待 ADC 转换结果
```

最后，当 AD_DONE 高电位时读取 ADC 结果：

```
WORD        Data;               //  两字节结果：放在 ADCRH 和 ADCRL
Data$1  =   ADCRH
Data$0  =   ADCRL;
Data    =   Data >> 4;          //  或 Data = (ADCRH << 8) | ADCRL;
```

ADC 也可以利用下面方法停用：

```
$   ADCC    Disable;
```

或

```
ADCC    =   0;
```

12.2.6. ADC 相关寄存器

12.2.6.1. ADC 数据高位寄存器(*ADCRH*)，地址 = 0x1E

位	初始值	读/写	描 述
7 - 0	-	只读	这 8 个只读位将是 AD 转换结果的 [11:4] 位。寄存器的第 7 位是任何分辨率 ADC 转换结果的 MSB。

12.2.6.2. ADC 数据低位寄存器(*ADCRL*)，地址 = 0x1F

位	初始值	读/写	描 述
7 - 4	-	只读	这 4 个只读位是 ADC 转换结果的位 [3:0]
3 - 0	-	-	保留

12.2.6.3. ADC 控制寄存器 (*ADCC*)，地址 = 0x20

位	初始值	读/写	描 述
7	0	读/写	启用 ADC 功能。0/1：停用/启用
6	-	读/写	ADC 转换过程控制位 写“1”开始 AD 转换，同时自动清零完成标志。 读到“1”表示完成 AD 的转换。读到“0”表示进行中。
5 - 2	0000	读/写	通道选择。 以下 4 位用来选择 AD 转换的输入信号： 0000: PA0/AD0 0001: PA1/AD1 0010: PA2/AD2 0011: PA3/AD3 0100: PA4/AD4 0101: PA5/AD5 0110: PA6/AD6 0111: PA7/AD7 1000-1100: 保留 1101: BEMF(COM 由 ZCPC[3]) 1110: VDD/4 1111: Bandgap 参考电压
1 - 0	-	-	保留（写 0）

12.2.6.4. ADC 模式寄存器（*ADCM*），地址 = 0x21

位	初始值	读/写	描 述
7	0	读/写	ADC 触发模式选择 0/1：命令/PWM 触发模式
6	0	读/写	中心模式 PWM 触发 ADC。0/1：向上/向下计数
5 - 4	-	-	保留
3 - 1	000	读/写	ADC 时钟源选择： 000：CLK（系统时钟）÷ 1 001：CLK（系统时钟）÷ 2 010：CLK（系统时钟）÷ 4 011：CLK（系统时钟）÷ 8 100：CLK（系统时钟）÷ 16 101：CLK（系统时钟）÷ 32 110：CLK（系统时钟）÷ 64 111：CLK（系统时钟）÷ 128
0	0	只写	ADC 参考电压选择. 0: VDD 1: Bandgap 电压

12.3. PWM 生成器触发 AD 转换

PWM 发生器还支持触发 AD 转换。它可以通过设置寄存器（ADCM.7）的触发使能位和 ADCC 寄存器（ADCC.7）的 ADC 使能位触发 AD 转换。当 PWM 计数器的计数与 PWMADCH 和 PWMADCL 寄存器的设置值相匹配时，将发出控制信号触发 ADC 启动，如图 29 所示。AD 转换完成后，触发使能位将自动清零。

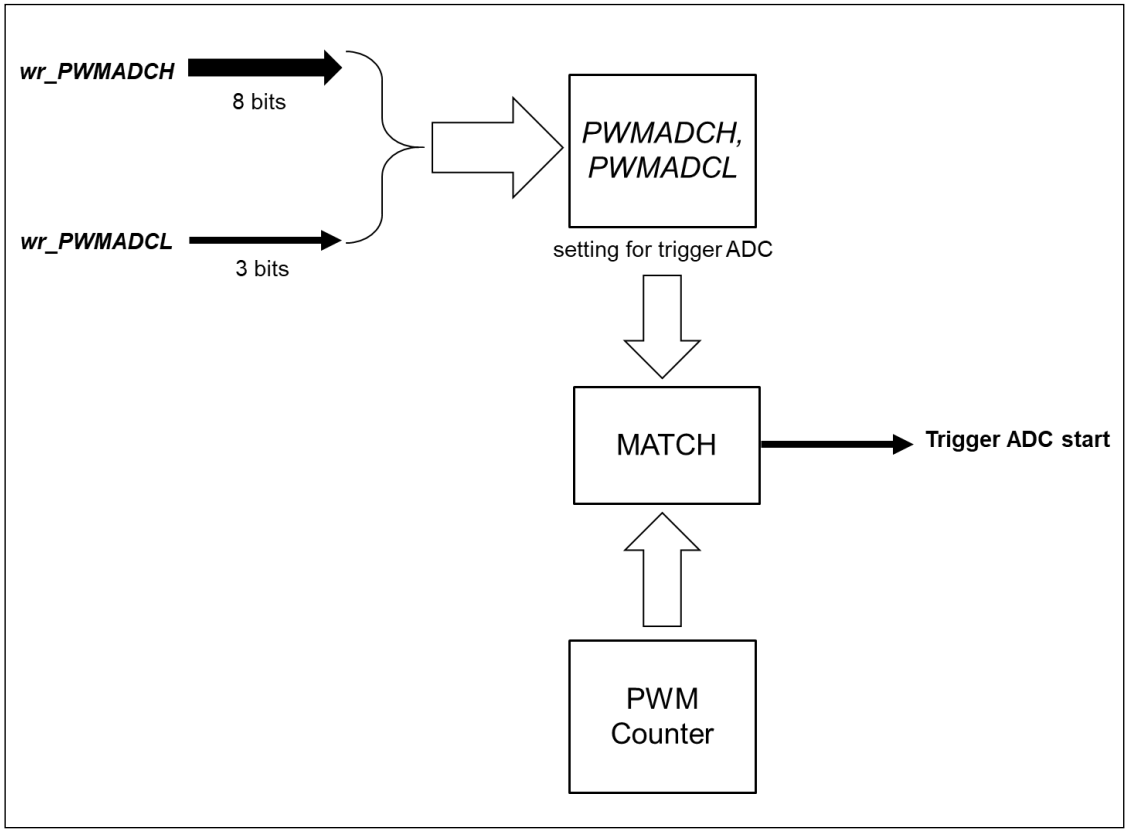


图 29：PWM 生成器触发 AD 转换框图

12.3.1. PWM 触发 ADC - PWM 计数器高电平寄存器 (*PWMADCH*), 地址= 0x41

位	初始值	读/写	描 述
7 - 0	8'h00	只写	触发 ADC - PWM 计数值位[11:4]

12.3.2. PWM 触发 ADC - PWM 计数器低电平寄存器 (PWMADCL), 地址= 0x42

位	初始值	读/写	描 述
7 - 5	3'h0	只写	触发 ADC - PWM 计数值位[3:1]
4 - 0	-	-	保留

12.4. 输入脉冲捕获

输入脉冲捕捉功能在需要频率和脉冲测量的应用中非常有用。图 30 显示了 MF324 中输入脉冲捕捉的硬件图，脉冲捕捉模块的时基可以是系统时钟或 IHRCX2，测量的输入信号可以是 PA3、PA4 或 PA6。

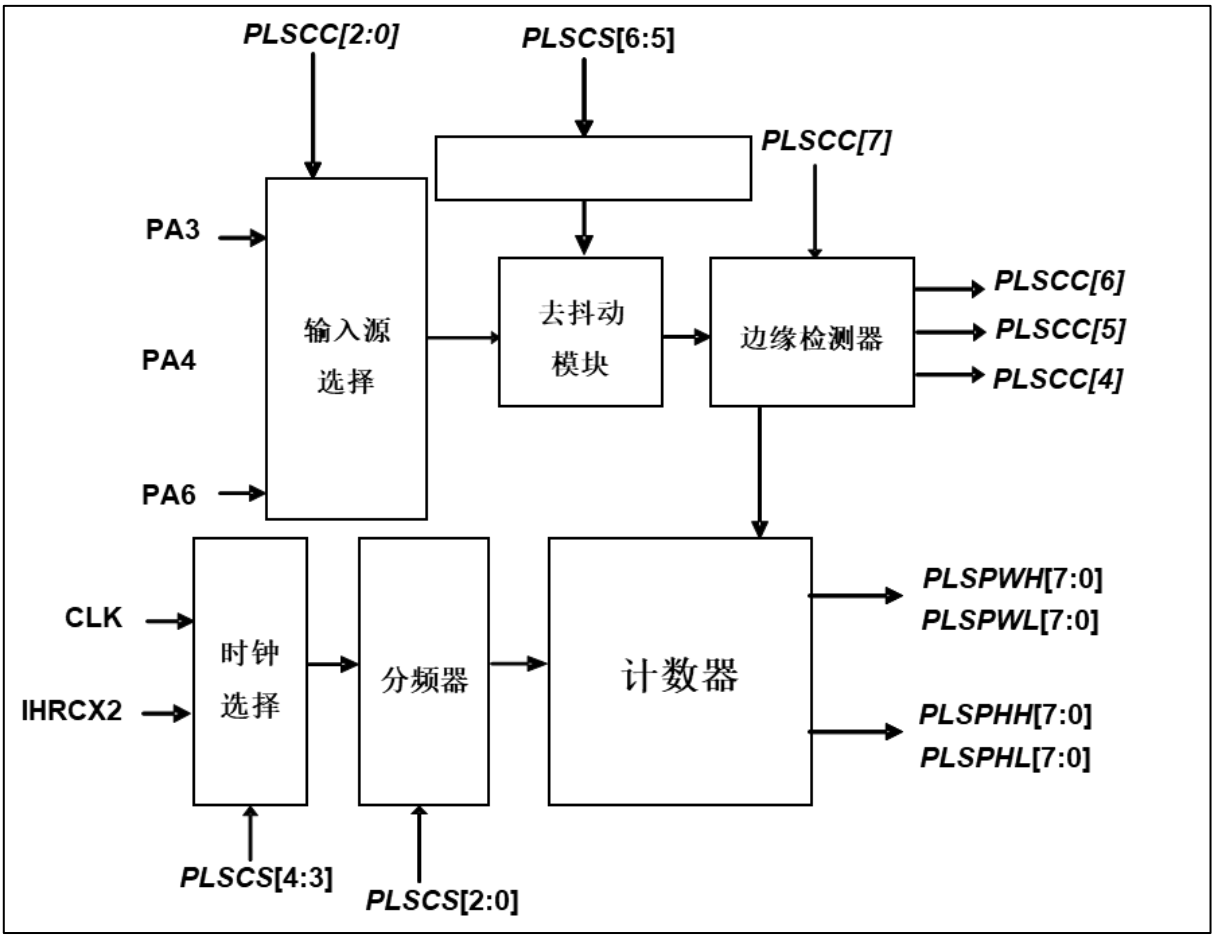


图 30：输入脉冲捕获的硬件框图

12.4.1. 脉冲捕获控制寄存器 (*PLSCC*)，地址 = 0x32

位	初始值	读/写	描 述
7	0	只写	0：停止进行脉冲捕捉 1：开始进行脉冲捕捉并清除脉冲捕捉溢出或完成标志
6		只读	脉冲捕捉溢出标志。在完成脉冲捕捉操作后自动清除该位。
5		只读	脉冲捕获完成标志
4		只读	全周期周期捕捉完成标志
3	1	读/写	0 / 1：捕捉低脉冲/高脉冲
2 - 0	000	读/写	脉冲捕获信号源 000: PA3 001: PA4 010: PA6 011-111：保留

12.4.2. 脉冲捕获分频寄存器(*PLSCS*)，地址 = 0x33

位	初始值	读/写	描 述
7	-	-	保留。请保持为 0。
6 - 5	00	读/写	脉冲捕获输入源去抖动时间选择 00：无 01：1 个脉冲捕获时钟 10：2 个脉冲捕获时钟 11：3 个脉冲捕获时钟
4:3	00	读/写	脉冲捕获时钟源 00：系统时钟 01：IHRC*2 1X：保留
2 - 0	000	读/写	脉冲捕获的时钟分频 000：/1 001：/2 010：/4 011：/8 100：/16 101：/32 110：/64 111：/128

12.4.3. 脉冲捕获脉宽高位寄存器 (*PLSPWH*)，地址 = 0x39

位	初始值	读/写	描 述
7 - 0	0xFF	读/写	R：脉冲捕获脉宽的高字节 W：脉冲捕捉溢出边界的高字节

12.4.4. 脉冲捕获脉宽低位寄存器 (PLSPWL)，地址 = 0x3A

位	初始值	读/写	描 述
7 - 0	0xFF	读/写	R：脉冲捕获脉宽的低字节 W：脉冲捕捉溢出边界的低字节

12.4.5. 脉冲捕获高准位脉宽高位寄存器 (*PLSPHH*)，地址 = 0x3B

位	初始值	读/写	描 述
7 - 0	-	只读	脉冲捕获高准位脉宽的高字节

12.4.6. 脉冲捕获高准位脉宽低位寄存器 (*PLSPHL*)，地址 = 0x3C

位	初始值	读/写	描 述
7 - 0	-	只读	脉冲捕获低准位脉宽的低字节

12.5. 特殊比较器

MF324 内置两个特殊比较器，可为三相无刷直流电机应用提供更好的支持。

12.5.1. 限流比较器

MF324 提供比较器电流检测功能。可比较 PA7 与内部基准电压之间的信号。若使用比较器功能，请通过设置 PADIER.7 寄存器关闭数字输入。内部基准电压可在 50mV 至 210mV 范围内以 10mV 为步进进行选择，具体请参阅 LCS[3:0]寄存器。

上电复位后比较器默认禁用，可通过设置 LCS.7=1 启用。

比较器结果可通过 LCS.5 读取，该功能同时具备去毛刺特性。更多细节请参阅寄存器 ZCPS[2:0]。

值得一提的是，该功能还具备关闭 PWM 的能力，可有效降低电机电流。使用此功能需启用 LCS.6。

12.5.2. 限流设置寄存器 (LCS), 地址= 0x2F

位	初始值	读/写	描 述
7	0	读/写	启用限值比较器。0 / 1：停用/启用
6	0	只写	启用极限电流比较器结果关闭 PWM 功能。0 / 1：停用 / 启用
5	0	读/写	限流比较器结果
4	0	读/写	ADC 参考电压。0/1：停用/启用 注意：若设置 LCS 启用，请将其打开
3- 0	0x05	读/写	限流比较器的内部参考电压 = 50 mV + (LCS[3:0] * 10 mV)

12.5.3. 过零点比较器

MF324 内置一路用于检测反电动势（BEMF）过零点（ZCP）的比较器，该比较器可实现三相反电动势中任意一相信号与零点的电压比较。其中，三相反电动势的目标检测相可通过 ZCPC 寄存器 [7:6] 位段进行选择；比较器的过零检测结果（ZCP_X，X 代表 A、B、C 三相中的任一相）可由 ZCPC.5 读取，同时可通过 ZCPC.2 配置结果信号为反向输出或缓冲输出模式，其消抖功能的具体参数需参考 ZCPS [5:3] 。此外，通过 ZCPC .3 ，还可切换比较器的输入参考电压为外部电压或内部过零点公共电压。

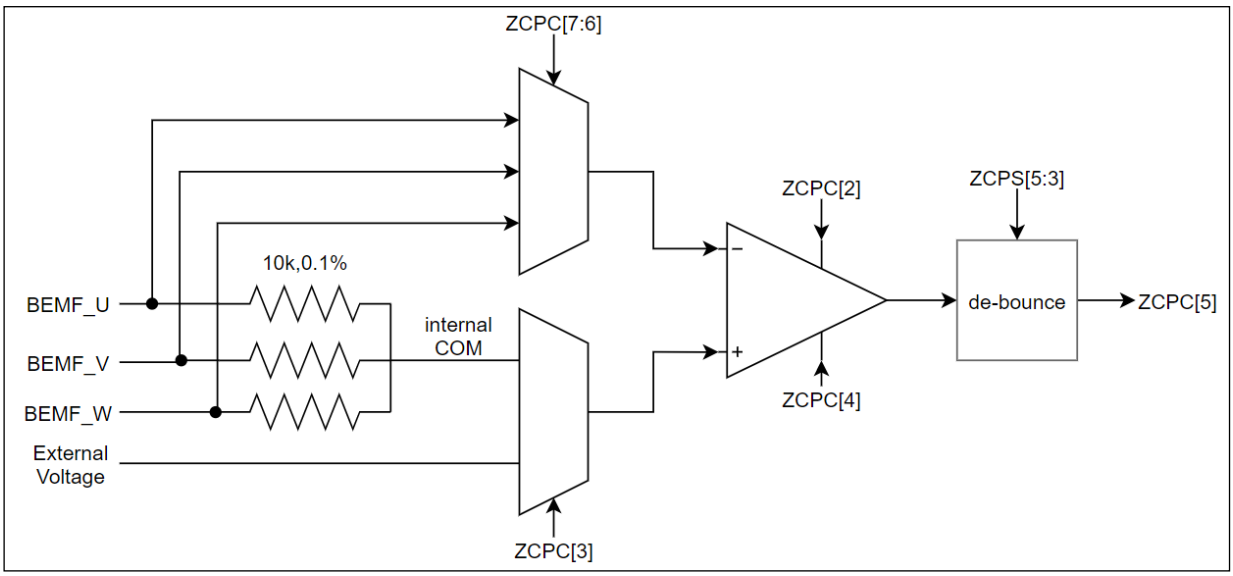


图 31：过零点比较器的硬件电路图

12.5.4. 过零点控制寄存器 (ZCPC)，地址= 0x30

位	初始值	读/写	描 述
7 - 6	00	读/写	启用并选择三相 BEMF 中的一相 00：PA0 相 BEMF 01：PA1 相 BEMF 10：PA2 相 BEMF 11： 停用
5	0	只读	过零点比较器结果。
4	0	读/写	过零点比较器结果数据速率 0: 2M 1: 4M 注：选择 2 M 数据速率时，比较器可保证 5 mV 的偏移精度；若选择 4M 速率，其偏移量会相应增大。
3	0	读/写	ZCP 比较器输入电压选择 0：采用内部 ZCP 公共电压 1：采用 PA5 引脚输入的外部电压
2	0	读/写	用于控制比较器结果的输出状态。 0 / 1：正相 / 反相
1 - 0	00	读/写	保留

12.5.5. 过零点设置寄存器 (ZCPS)，地址= 0x31

位	初始值	读/写	描 述
7	0	读/写	限流比较器结果数据速率。 0: 2M 1: 4M 注：选择 2M 数据速率时，比较器可保证 5mV 的偏移精度；若选择 4 M 速率，其偏移量会相应增大。
6	-	-	保留，写 0。
5 - 3	000	只写	消除 ZCP 比较器结果的抖动。 000: 关闭 001: 1 us 010: 2 us 011: 4 us 100: 8 us 101: 16 us 11X: 保留
2 - 0	000	只写	消除 LC 比较器结果的抖动。 000: 关闭 001: 1 us 010: 2 us 011: 4 us 100: 8 us 101-111: 保留

12.6. 乘法器

12.6.1. 8×8 乘法器

芯片内置一 8x8 乘法器以加强硬件的运算功能。这个乘法运算方式是 8x8 的无符号运算并且在 9 个时钟周期内完成运算。在设置开始位(*EARITH*.6)之前，乘数与被乘数都要放在寄存器 *M8OP1H*/*M8OP1L* 和 *M8OP2* 上。完成 8x8 后，内部设置完成位 (*EARITH*.1)，运算结果会放在寄存器 M8RS1/M8RS0 上。乘法器的硬件框图如图 32 所示。

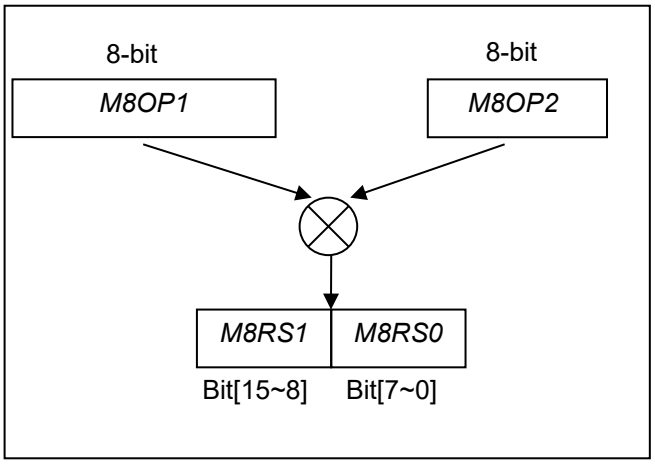


图 32：硬件乘法器框图

12.6.2. 算术运算寄存器 (*EARITH*)，地址 = 0x1D

位	初始值	读/写	描 述
7	-	-	保留
6	-	只写	8X8 乘法器开始位，此位由硬件清零
5	-	-	保留
4 - 3	-	-	保留
2	-	-	保留
1	-	只读	8X8 乘法器完成标志，该位由开始位清零
0	-	-	保留

12.6.3. 8X8 乘法器运算对象 1 寄存器(*M8OP1*)，地址 = 0x35

位	初始值	读/写	描 述
7 - 0	-	只写	乘法运算的运算对象 1

12.6.4. 8X8 乘法器运算对象 2 寄存器(*M8OP2*)，地址 = 0x36

位	初始值	读/写	描 述
7 - 0	-	只写	乘法运算的运算对象 2

12.6.5. 8X8 乘法器结果 1 寄存器(*M8RS1*)，地址 = 0x37

位	初始值	读/写	描 述
7 - 0	-	只读	乘法运算的结果位 15~8

12.6.6. 8X8 乘法器结果 0 寄存器(*M8RS0*)，地址 = 0x38

位	初始值	读/写	描 述
7 - 0	-	只读	乘法运算的结果位 7~0

13. 烧录方法

MF324 的烧录脚为：PA3, PA5, PA6, VDD 和 GND 这 5 个引脚。

请使用 5S-P-003x 进行烧录。3S-P-002x 或之前的烧录器皆不支持烧录 MF324。

应用范围:

- 单独封装 IC，并在烧录器的 IC 插座或连接分选机烧录。
- 合封（MCP）IC，但与 MF324 合封的 IC 及元件不会被以下电压破坏，也不会钳制以下电压的产生。

普通烧录模式电压条件：

(1) VDD 等于 5.5V，而最大供给电流最高可达约 20mA。

(2) 其他烧录引脚（GND 除外）等于 VDD。

用户应确认在 MCP 或在板烧录模式下使用本产品时，外围组件或电路不会因上述电压而损坏，且不会产生上述电压。

重要注意事项：

- 如在 handler 上对 IC 进行烧录，请务必按照 APN004 及 APN011 的指示进行。
- 为对抗烧录时的杂讯干扰，请于烧录时在分选机连接 IC 连接器一端的 VDD 和 GND 之间连接 0.01uF 电容。但切忌连接标值 0.01uF 以上的电容，以免影响普通烧录模式的运行。

使用 5S-P-003x 烧录 MF324，使用 Jumper7 转接程序信号。信号的连接取决于 IC 封装。请参阅 Writer 用户手册的第 5 章，为目标 IC 封装制作 Jumper7 转接板。

用户可以从以下网页链接获取用户手册。

http://www.padauk.com.tw/en/technical/index.aspx?kind=27

以 QFN-16 封装的 JP7 烧录连接为例。如图 33 所示，其烧录引脚分别为引脚 4（接地 GND）、引脚 6（PA5）、引脚 9（PA3）、引脚 11（电源 VDD）以及引脚 13（PA6）。JP7 跳线的具体连接方式如下。

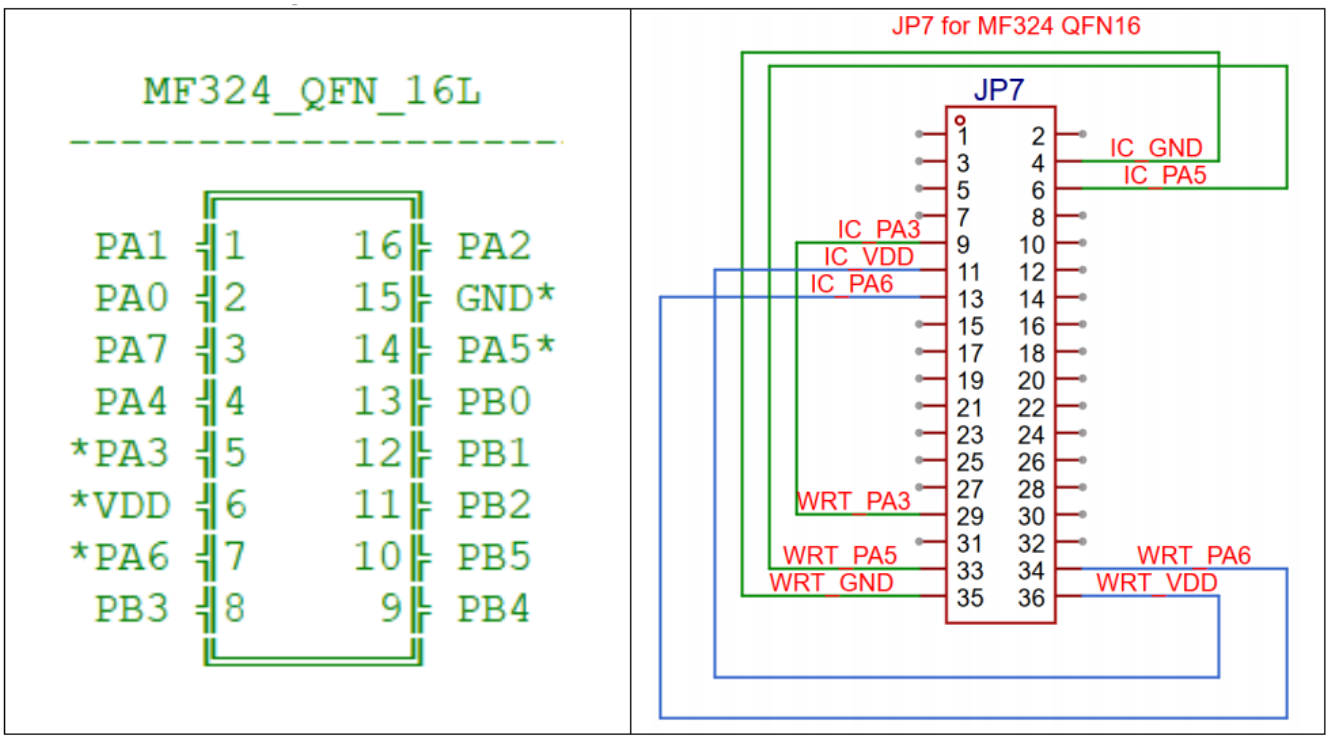


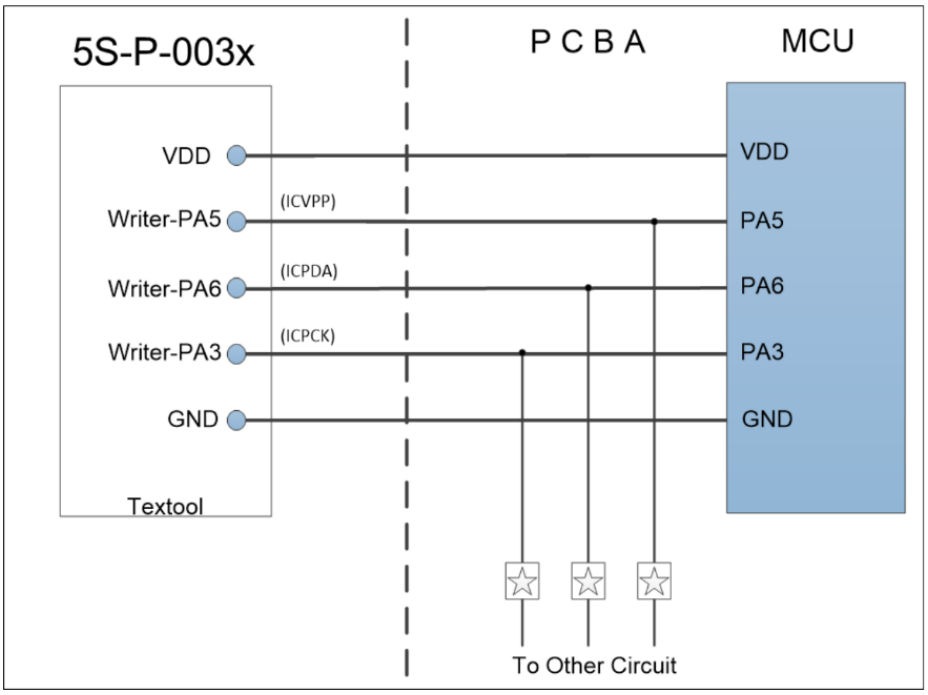
图 33：使用 P003x 烧录 QFN-16 封装的 Jumper7 跳线图

在烧录器上安装好 JP7 并下载 PDK 文件后，依次执行以下操作：点击 <Convert> ➔ <To Package> ➔ 选择要下载的 PDK ➔ 选择封装 ➔ 保存 PDK 文件

在线烧录 (On-board Writing)

MF324 支持在线烧录（On-board Writing）。在线烧录是指在芯片本身以及其他外围电路和器件已经集成/焊接在 PCB（印制电路板）上之后，对芯片进行编程的工况。

在线烧录需要使用 5S-P-003x 的五根信号线：ICPCK、ICPDA、VDD、GND 和 ICVPP。它们分别对应连接到芯片的 PA3、PA6、VDD、GND 和 PA5 引脚。



上图展示了 MF324 在线烧录（On-board Writing）的连接方式。在此图中，标记为「☆」的组件可以是电阻或电容。它们用于将烧录信号线与外围电路进行隔离。若使用电阻，其阻值应 ≧ 10KΩ；若使用电容，其容值应 ≦ 220pF。

注意事项:

- 任何≦5.0V 的齐纳二极管（稳压二极管），或任何会导致产生 5.0V 钳位电压的电路，切勿连接在 PCB 的 VDD 与 GND 之间。
- 任何≧500uF 的电容，切勿连接在 PCB 的 VDD 与 GND 之间。
- 通常情况下，烧录信号引脚（PA3、PA5 和 PA6）不应被视作强输出引脚。

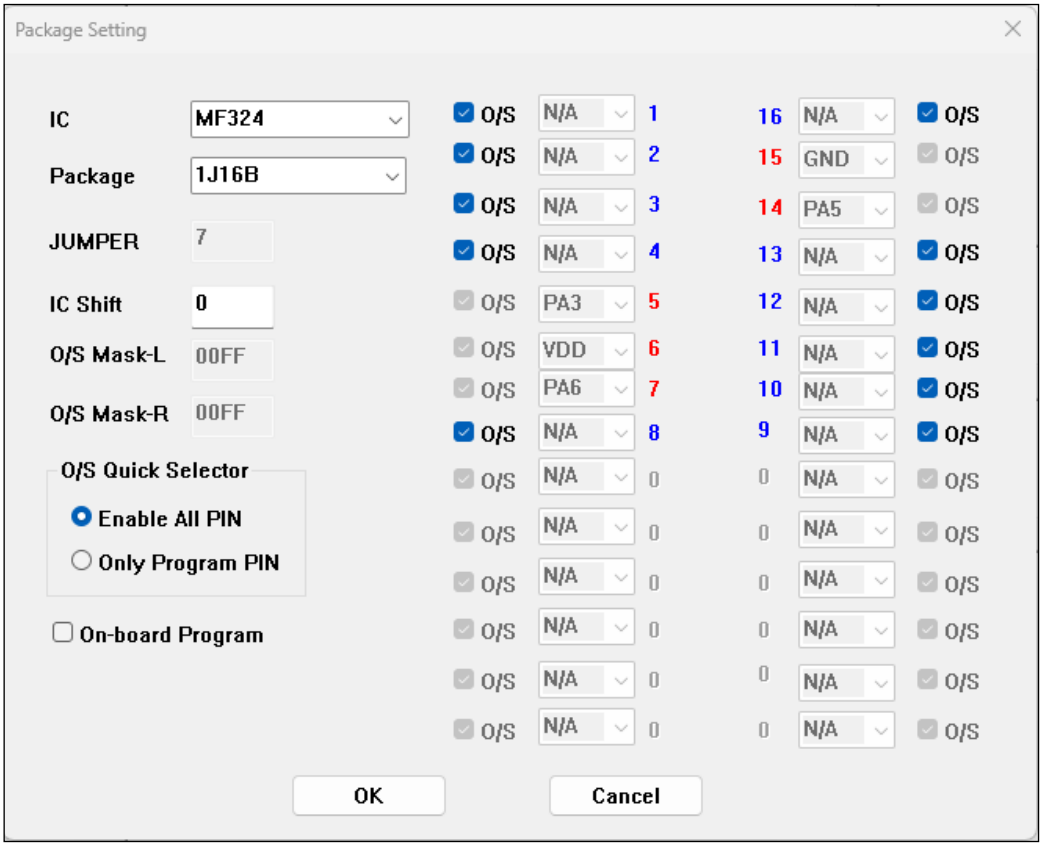


图.34: IDE 封装配置图形界面

从图形用户界面加载 PDK，插入 JP7，然后在烧录座上插入 IC，无需移位。LCDM 显示 IC 就绪后，即可烧录。

14. 指令

符号	描 述
ACC	累加器（Accumulator 的缩写）
a	累加器（Accumulator 在程序里的代表符号）
sp	堆栈指针
flag	标志寄存器
I	即时数据
&	逻辑 AND
|	逻辑 OR
←	移动
^	异或 OR
+	加
—	减
-～	NOT（逻辑补数，1 补数）
∓	2 补数
OV	溢出（2 补数系统的运算结果超出范围）
Z	零（如果零运算单元操作的结果是 0，这位设置为 1）
C	进位(Carry) 在无符号数系统中，执行加法时可能产生进位，执行减法时可能需要借位。
AC	辅助进位标志(Auxiliary Carry)。 如果在 ALU 运算结果中低位半字节产生了进位，则该位被置为 1。
pc0	FPPA0 的程序计数器
pc1	FPPA1 的程序计数器
pc2	FPPA2 的程序计数器
pc3	FPPA3 的程序计数器
pc4	FPPA4 的程序计数器
pc5	FPPA5 的程序计数器
pc6	FPPA6 的程序计数器
pc7	FPPA7 的程序计数器

©Copyright 2026, PADAUK Technology Co. Ltd

13.1. 指令表

指令	功能	周期	Z	C	AC	OV
数据传输类指令						
mov a, I	*mov* a, 0x0f; a ← 0fh;	1	-	-	-	-
mov M, a	*mov* MEM, a; MEM ← a	1	-	-	-	-
mov a, M	*mov* a, MEM ; a ← MEM; 当 MEM 为零时，标志位 Z 会被置位.	1	Y	-	-	-
mov a, IO	*mov* a, pa ; a ← pa; 当 pa 为零时，标志位 Z 会被置位	1	Y	-	-	-
mov IO, a	*mov* pb, a; pb ← a ;	1	-	-	-	-
nmov M, a	*nmov* MEM, a; MEM ← ∓a	1	-	-	-	-
nmov a, M	*mov* a, MEM ; a ← ∓MEM; Flag Z is set when ∓MEM is zero.	1	-	-	-	-
ldtabh index	*ldtabh index;* a ← {bit 15~8 of MTP [index]};	**2**	-	-	-	-
ldtabl index	*ldtabl index;* a ← {bit7~0 of MTP [index]};	**2**	-	-	-	-
ldt16 word	*ldt16* word; word ← 16-bit timer	1	-	-	-	-
stt16 word	*stt16* word; 16-bit timer ← word	1	-	-	-	-
idxm a, index	*idxm* a, index; a ← [index], where index is declared by word.	**2**	-	-	-	-
idxm index, a	*idxm* index, a; [index] ← a; where index is declared by word.	**2**	-	-	-	-
xch M	*xch* MEM ; MEM ← a , a ← MEM	1	-	-	-	-
pushaf	*pushaf*; [sp] ← {flag, ACC}; sp ← sp + 2 ;	1	-	-	-	-
popaf	*popaf;* sp ← sp - 2 ; {Flag, ACC} ← [sp] ;	1	Y	Y	Y	Y
pushw word	*pushw word*; [sp] ← *word* ; sp ← sp + 2	**2**	-	-	-	-
popw word	*popw word*; sp ← sp - 2 ; word ← [sp] ;	**2**	-	-	-	-

指令	功能	周期	Z	C	AC	OV
算术运算类指令						
add a, I	*add* a, 0x0f ; a ← a + 0fh	1	Y	Y	Y	Y
add a, M	*add* a, MEM ; a ← a + MEM	1	Y	Y	Y	Y
add M, a	*add* MEM, a; MEM ← a + MEM	1	Y	Y	Y	Y
addc a, M	*addc* a, MEM ; a ← a + MEM + C	1	Y	Y	Y	Y
addc M, a	*addc* MEM, a ; MEM ← a + MEM + C	1	Y	Y	Y	Y
addc a	*addc* a ; a ← a + C	1	Y	Y	Y	Y
addc M	*addc* MEM ; MEM ← MEM + C	1	Y	Y	Y	Y
nadd a, M	*nadd* a, MEM ; a ← ∓a + MEM	1	Y	Y	Y	Y
nadd M, a	*nadd* MEM, a; MEM ← ∓MEM + a	1	Y	Y	Y	Y
sub a, I	*sub* a, 0x0f; a ← a - 0fh (a + [2’s complement of 0fh])	1	Y	Y	Y	Y
sub a, M	*sub* a, MEM ; a ← a - MEM (a + [2’s complement of M])	1	Y	Y	Y	Y
sub M, a	*sub* MEM, a; MEM ← MEM - a (MEM + [2’s complement of a])	1	Y	Y	Y	Y
subc a, M	*subc* MEM, a ; a ← a – MEM - C	1	Y	Y	Y	Y
subc M, a	*subc* MEM, a ; MEM ← MEM – a - C	1	Y	Y	Y	Y
subc a	*subc* a; a ← a - C	1	Y	Y	Y	Y
subc M	*subc* MEM; MEM ← MEM - C	1	Y	Y	Y	Y
inc M	*inc* MEM ; MEM ← MEM + 1	1	Y	Y	Y	Y
dec M	*dec* MEM; MEM ← MEM - 1	1	Y	Y	Y	Y
clear M	*clear* MEM ; MEM ← 0	1	-	-	-	-

指令	功能	周期	Z	C	AC	OV
移位运算类指令						
sr a	*sr* a ; a (0,b7,b6,b5,b4,b3,b2,b1) ← a (b7,b6,b5,b4,b3,b2,b1,b0), C ← a(b0)	1	-	Y	-	-
src a	*src* a ; a (c,b7,b6,b5,b4,b3,b2,b1) ← a (b7,b6,b5,b4,b3,b2,b1,b0), C ← a(b0)	1	-	Y	-	-
sr M	*sr* MEM ; MEM(0,b7,b6,b5,b4,b3,b2,b1) ← MEM(b7,b6,b5,b4,b3,b2,b1,b0), C ← MEM(b0)	1	-	Y	-	-
src M	*src* MEM ; MEM(c,b7,b6,b5,b4,b3,b2,b1) ← MEM (b7,b6,b5,b4,b3,b2,b1,b0), C ← MEM(b0)	1	-	Y	-	-
sl a	*sl* a ; a (b6,b5,b4,b3,b2,b1,b0,0) ← a (b7,b6,b5,b4,b3,b2,b1,b0), C ← a (b7)	1	-	Y	-	-
slc a	*slc* a ; a (b6,b5,b4,b3,b2,b1,b0,c) ← a (b7,b6,b5,b4,b3,b2,b1,b0), C ← a(b7)	1	-	Y	-	-
sl M	*sl* MEM ; MEM (b6,b5,b4,b3,b2,b1,b0,0) ← MEM (b7,b6,b5,b4,b3,b2,b1,b0), C ← MEM(b7)	1	-	Y	-	-
slc M	*slc* MEM ; MEM (b6,b5,b4,b3,b2,b1,b0,C) ← MEM (b7,b6,b5,b4,b3,b2,b1,b0), C ← MEM (b7)	1	-	Y	-	-
swap a	*swap* a ; a (b3,b2,b1,b0,b7,b6,b5,b4) ← a (b7,b6,b5,b4,b3,b2,b1,b0)	1	-	-	-	-
swap M	*swap* MEM ; MEM (b3,b2,b1,b0,b7,b6,b5,b4) ← MEM (b7,b6,b5,b4,b3,b2,b1,b0)	1	-	-	-	-
逻辑运算类指令						
and a, I	*and* a, 0x0f ; a ← a & 0fh	1	Y	-	-	-
and a, M	*and* a, RAM10 ; a ← a & RAM10	1	Y	-	-	-
and M, a	*and* MEM, a ; MEM ← a & MEM	1	Y	-	-	-
or a, I	*or* a, 0x0f ; a ← a | 0fh	1	Y	-	-	-
or a, M	*or* a, MEM ; a ← a | MEM	1	Y	-	-	-
or M, a	*or* MEM, a ; MEM ← a | MEM	1	Y	-	-	-
xor a, I	*xor* a, 0x0f ; a ← a ^ 0fh	1	Y	-	-	-
xor IO, a	*xor* *pa, a ;* pa ← a ^ pa ;	1	-	-	-	-
xor a, M	*xor* a, MEM ; a ← a ^ RAM10	1	Y	-	-	-
xor M, a	*xor* MEM, a ; MEM ← a ^ MEM	1	Y	-	-	-
not a	*not* a ; a ← ~a	1	Y	-	-	-
not M	*not* MEM ; MEM ← ~MEM	1	Y	-	-	-
neg a	*neg* a; a ← ∓a	1	Y	-	-	-
neg M	*neg* MEM; MEM ← ∓MEM	1	Y	-	-	-
comp a, I	*comp* a, 0x55; 等效于(a - 0x55)，并改变标志位	1	Y	Y	Y	Y
comp *a, M*	*comp* a,MEM;等效于(a - MEM)，并改变标志位 Flag	1	Y	Y	Y	Y
comp *M, a*	*comp* MEM,a; 等效于(MEM - a)，并改变标志位 Flag	1	Y	Y	Y	Y

指令	功能	周期	Z	C	AC	OV
位运算类指令						
set0 IO.n	*set0* pa.5 ; PA5=0	1	-	-	-	-
set1 IO.n	*set1* pb.5 ; PB5=1	1	-	-	-	-
set0 M.n	*set0* MEM.5 ; set bit 5 of MEM to low	1	-	-	-	-
set1 M.n	*set1* MEM.5 ; set bit 5 of MEM to high	1	-	-	-	-
swapc *IO.n*	*swapc* IO.0; C ← IO.0 , IO.0 ← C 当 IO.0 是输出脚位，进位标志 C 将被送到 IO.0 脚 当 IO.0 是输入脚位，IO.0 脚的状态将被送到进位标志 C	1	-	Y	-	-
条件运算类指令						
ceqsn a, I	*ceqsn* a, 0x55 ; *inc* MEM ; *goto* error ; 假如 a=0x55, then “goto error”; 否则, “inc MEM”.	1	Y	Y	Y	Y
ceqsn a, M	*ceqsn* a, MEM; 假如 a=MEM，跳过下一个指令	1	Y	Y	Y	Y
ceqsn M, a	*ceqsn* MEM, a; 假如 a=MEM，跳过下一个指令	1	Y	Y	Y	Y
cneqsn a, M	*cneqsn* a, MEM; 假如 a≠MEM，跳过下一个指令	1	Y	Y	Y	Y
cneqsn M, a	*cneqsn* MEM,a; 假如 a≠MEM，跳过下一个指令	1	Y	Y	Y	Y
cneqsn a, I	*cneqsn* a, 0x55 ; *inc* MEM ; *goto* error ; 假如 a≠0x55, then “goto error”; 否则, “inc MEM”.	1	Y	Y	Y	Y
t0sn IO.n	*t0sn* pa.5;如果 PA5 是 0，跳过下一个指令	1	-	-	-	-
t1sn IO.n	*t1sn* pa.5; 如果 PA5 是 1，跳过下一个指令	1	-	-	-	-
t0sn M.n	*t0sn* MEM.5 ; 如果 MEM 的位 5 是 0，跳过下一个指令	1	-	-	-	-
t1sn M.n	*t1sn* MEM.5 ; 如果 MEM 的位 5 是 1，跳过下一个指令	1	-	-	-	-
izsn a	*izsn* a; a ← a + 1,若 a=0，跳过下一个指令	1	Y	Y	Y	Y
dzsn a	*dzsn* a; a ← a – 1，若 a=0，跳过下一个指令	1	Y	Y	Y	Y
izsn M	*izsn* MEM; MEM ← MEM + 1，若 MEM=0，跳过下一个指令	1	Y	Y	Y	Y
dzsn M	*dzsn* MEM; MEM ← MEM - 1，若 MEM=0，跳过下一个指令	1	Y	Y	Y	Y

指令	功能	周期	Z	C	AC	OV
系统控制类指令						
call label	*call* function1; [sp] ← pc + 1, pc ← function1, sp ← sp + 2	**1**	-	-	-	-
goto label	*goto* routine1; 跳到 routine1 并继续执行程序	**1**	-	-	-	-
delay I	*delay* 0x05; 在此延迟 6 个周期	1	-	-	-	-
delay a	*delay* a; 假如 ACC=0fh，在此延迟 16 个周期	1	-	-	-	-
delay M	*delay* M; 假如 M=ffh，在此延迟 256 个周期	1	-	-	-	-
***delay* 指令的注意事项：** (1) 由于 ACC 是指令计数时的暂时缓冲区，请确保执行此指令时不会被中断。否则，延时时间可能不是预期的。 (2) 单一 FPPA 模式下不支持此指令。						
ret I	*ret* 0x55; A ← 55h ret ;	**2**	-	-	-	-
ret	*ret;* sp ← sp - 2 pc ← [sp]	**2**	-	-	-	-
reti	*reti*; 从中断服务程序返回到原程序 在这指令执行之后，全局中断将自动启用	**2**	-	-	-	-
nop	*nop*; 没任何改变	1	-	-	-	-
pcadd a	*pcadd* *a*; pc ← pc + a	**2**	-	-	-	-
engint	*engint*; 中断请求可送到 FPPA0	1	-	-	-	-
disgint	*disgint* ; 送到 FPPA0 的中断请求全部被挡住	1	-	-	-	-
stopsys	*stopsys*; 停止系统时钟和关闭系统	1	-	-	-	-
stopexe	*stopexe*; 停止系统时钟，但时钟源还是保持原来的状态	1	-	-	-	-
reset	*reset*; 复位整个单片机	1	-	-	-	-
wdreset	*wdreset* ; 复位看门狗定时器	1	-	-	-	-
pmode n	各 FPPA 单元的操作模式选择 *pmode* 0; 设置 FPPA 单元的带宽共享模式为模式 0	1	-	-	-	-

指令	功能	周期	Z	C	AC	OV
系统控制类指令						
pmode n	各 FPPA 单元的操作模式选择 *pmode* 0; 设置 FPPA 单元的带宽共享模式为模式 0 模式 FPPA0 ~ FPPA7 带宽共享 0: /2, /2 1: /2, /4, /4 2: /4, /2, /4 3: /2, /4, /8, /8 4: /4, /2, /8, /8 5: /8, /2, /4, /8 6: /4, /4, /4, /4 7: /8, /4, /4, /4, /8 8: /2, /8, /8, /8, /8 9: /4, /4, /4, /8, /8 10: /8, /2, /8, /8, /8 11: /2, /8, /8, /8, /16, /16 12: /16, /2, /8, /8, /8, /16 13: /4, /4, /8, /8, /8, /8 14: /8, /4, /4, /8, /8, /8 15: /4, /4, /4, /8, /16, /16 16: /8, /4, /4, /4, /16, /16 17: /16, /4, /4, /4, /8, /16 18: /2, /8, /8, /16, /16, /16, /16 19: /8, /2, /8, /16, /16, /16, /16 20: /16, /2, /8, /8, /16, /16, /16 21: /4, /4, /4, /16, /16, /16, /16 22: /16, /4, /4, /4, /16, /16, /16 23: /4, /8, /8, /8, /8, /8, /8 24: /8, /2, /16, /16, /16, /16, /16, /16 25: /4, /8, /4, /8, /16, /16, /16, /16 26: /8, /4, /4, /8, /16, /16, /16, /16 27: /2, /8, /16, /16, /16, /16, /16, /16 28: /4, /4, /8, /8, /16, /16, /16, /16 29: /16, /2, /8, /16, /16, /16, /16, /16 30: /8, /4, /4, /8, /16, /16, /16, /16 31: /8, /8, /8, /8, /8, /8, /8, /8	1	-	-	-	-